SYNOPSYS®

VirSim User Guide

Version 4.3

About this Manual

Contents

Chapter 1 — VirSim Overview

Chapter 2 — Starting VirSim

Chapter 3 — Hierarchy Browser

Chapter 4 — Waveform Window

Chapter 5 — Source Window

Chapter 6 — Logic Browser

Chapter 7 — Register Window

Chapter 8 — Interactive Window

Chapter 9 — Project Window

Chapter 10 — Time Units

Chapter 11 — Radices

Chapter 12 — Event Origin

Chapter 13 — Building Buses

Chapter 14 — Markers

Chapter 15 — Expressions

Chapter 16 — VirSim Setup

Chapter 17 — VCD+ (vpd) File Generation

Chapter 18 — Translating VCD and VCD+

Chapter 19 — Viewing OpenVera Assertions

Appendix — Waveform Defaults

Index

VirSim
User Guide

Version 4.3-2003.01, January 2003

Comments?
E-mail your comments about Synopsys documentation to doc@synopsys.com

SYNOPSYS®

Copyright Notice and Proprietary Information

Copyright © 2003 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Arcadia, C Level Design, C2HDL, C2V, C2VHDL, Calaveras Algorithm, CoCentric, COSSAP, CSim, DelayMill, DesignPower, DesignWare, Device Model Builder, Enterprise, EPIC, Formality, HSPICE, Hypermodel, I, InSpecs, in-Sync, LEDA, MAST, Meta, Meta-Software, ModelAccess, ModelExpress, ModelTools, PathBlazer, PathMill, PowerArc, PowerMill, PrimeTime, RailMill, Raphael, RapidScript, Saber, SmartLogic, SNUG, SolvNet, Stream Driven Simulator, Superlog, System Compiler, TestBench Manager, Testify, TetraMAX, TimeMill, TMA, VERA, VeriasHDL, and WaveCalc are registered trademarks of Synopsys, Inc.

Trademarks (™)

Active Parasitics, AFGen, Apollo, Apollo II, Apollo-DPII, Apollo-GA, ApolloGAII, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanTestchip, AvanWaves, BCView, Behavioral Compiler, BOA, BRT, Cedar, ChipPlanner, Circuit Analysis, Columbia, Columbia-CE, Comet 3D, Cosmos, Cosmos SE, CosmosLE, Cosmos-Scope, Cyclelink, Davinci, DC Expert, DC Expert *Plus*, DC Professional, DC Ultra, DC Ultra Plus, Design Advisor, Design Analyzer, Design Compiler, DesignerHDL, DesignTime, DFM-Workbench, DFT Compiler SoCBIST, Direct RTL, Direct Silicon Access, DW8051, DWPCI, Dynamic-Macromodeling, Dynamic Model Switcher, ECL Compiler, ECO Compiler, EDAnavigator, Encore, Encore PQ, Evaccess, ExpressModel, Floorplan Manager, Formal Model Checker, FormalVera, FoundryModel, FPGA Compiler II, FPGA *Express*, Frame Compiler, Frameway, Gatran, HDL Advisor, HDL Compiler, Hercules, Hercules-Explorer, Hercules-II, Hierarchical Optimization Technology, High Performance Option, HotPlace, HSPICE-Link, Integrator, Interactive Waveform Viewer, iQBus, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, JVXtreme, Liberty, Libra-Passport, Library Compiler, Libra-Visa, LRC, Mars, Mars-Rail, Mars-Xtalk, Medici, Metacapture, Metacircuit, Metamanager, Metamixsim, Milkyway, ModelSource, Module Compiler, MS-3200, MS-3400, NanoSim, Nova Product Family, Nova-ExploreRTL, Nova-Trans, Nova-VeriLint, Nova-VHDLlint, OpenVera, Optimum Silicon, Orion_ec, Parasitic View, Passport, Physical Compiler, Planet, Planet-PL, Planet-RTL, Polaris, Polaris-CBS, Polaris-MT, Power Compiler, PowerCODE, PowerGate, ProFPGA, Progen, Prospector, Proteus OPC, Protocol Compiler, PSMGen, Raphael-NES, RoadRunner, RTL Analyzer, Saber Co-Simulation, Saber for IC Design, SaberDesigner, SaberGuide, SaberRT, SaberScope, SaberSketch, Saturn, ScanBand, Schematic Compiler, Scirocco, Scirocco-i, Shadow Debugger, Silicon Blueprint, Silicon Early Access, SinglePass-SoC, Smart Extraction, SmartLicense, SmartModel Library, Softwire, Source-Level Design, Star, Star-DC, Star-Hspice, Star-HspiceLink, Star-MS, Star-MTB, Star-Power, Star-Rail, Star-RC, Star-RCXT, Star-Sim, Star-Sim XT, Star-Time, Star-XP, SWIFT, Taurus, Taurus-Device, Taurus-Layout, Taurus-Lithography, Taurus-OPC, Taurus-Process, Taurus-Topography, Taurus-Visual, Taurus-Workbench, Test Compiler, TestGen, TetraMAX TenX, The Power in Semiconductors, TheHDL, TimeSlice, TimeTracker, Timing Annotator, TopoPlace, TopoRoute, Trace-On-Demand, True-Hspice, TSUPREM-4, TymeWare, VCS, VCS Express, VCSi, Venus, Verification Portal, VFormal, VHDL Compiler, VHDL System Simulator, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (^{SM})

DesignSphere, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
All other product or company names may be trademarks of their respective owners.
Printed in the U.S.A.

Printed in the U.S.A.

Document Order Number: 37568-000 PA
VirSim User Guide, v4.3-2003.01

Contents

1. VirSim Overview

Operation Modes 1-2
Post Simulation Mode 1-2
Interactive Mode 1-3
Advantages of VirSim 1-3
Key Terms and Concepts 1-5
VirSim Windows 1-7
Interactive Window 1-8
Hierarchy Browser 1-9
Waveform Window 1-11
Register Window 1-12
Source Window 1-13
Logic Browser 1-14
Project Window 1-15
VirSim Navigation 1-16
Using Mouse Buttons 1-16
Selecting Signals 1-18

Drag-and-drop Operation 1-19
Linking Windows 1-21
Accelerator Keys 1-23

2. Starting VirSim

VCS VirSim GUI Startup Procedures 2-2
Opening VirSim in Interactive Mode 2-2
Creating Value Change Data (VPD) for Post Processing 2-3
Compile Options for Creating a VPD File from VCS 2-4
Starting VirSim in Post-Processing Mode 2-5
Options for Starting VirSim in Post Processing Mode 2-6

Scirocco VirSim GUI Startup Procedures 2-7
Starting Scirocco VirSim Example 2-7
VHDL Command Line Syntax 2-8
Scirocco VirSim Command Line Arguments 2-9

Standalone VirSim GUI Startup Procedures 2-11
Starting Standalone VirSim Example 2-11
Standalone VirSim Command Line Syntax 2-12
Standalone VirSim Command Line Arguments 2-13

Opening and Closing History Files 2-16
File Types and File Designators 2-16
Opening the Open File Dialog 2-17
Open File Dialog Display Area Descriptions 2-17
Opening a Single History File 2-22
Opening Multiple History Files 2-23
Closing History Files 2-23

Reopening History Files. 2-25
Saving and Loading a Configuration File. 2-26
Saving a Configuration File 2-27
Loading a Configuration File 2-28
Configuration File Format 2-29
Nested Configuration Files. 2-30
Incremental Loading of Configuration Files 2-30
Loading Sources 2-31

3. Hierarchy Browser

Introducing the Hierarchy Browser 3-2
Selecting a Design 3-3
Choosing View Options 3-4
Navigating the Hierarchy 3-6
Searching for Scopes and Signals 3-7
Filtering Signals Displayed on the Signals Tab 3-8
Adding Signals to a Group 3-9
Toolbar and Menu Reference 3-10
Toolbar 3-10
Context Sensitive Menus 3-11
Hierarchy Pane Context Sensitive Menu 3-11
Signal Pane Context-Sensitive Menu 3-12
Search Pane Context-Sensitive Menu 3-12
Menu Bar 3-13

File Menu 3-13
Edit Menu 3-14
Display Menu 3-14
View Menu 3-15

4. Waveform Window

Introducing the Waveform Window 4-2
Signal Name Pane 4-4
Signal Value Pane 4-4
Waveform Pane 4-6
Cursor Pane 4-7
Status Bar 4-8

Selecting and Viewing Signals 4-9
Adding and Reordering Signals 4-9
Adding Signals to a Signal Group 4-9
Reordering Signals in a Signal Group 4-10
Editing Signal Names and Bit Ranges 4-10
Using Multiple Signal Groups to View Relevant Data 4-11
Manually Creating a Signal Group 4-12
Automatically Creating AutoGroups 4-13
Switching Between Multiple Signal Groups 4-13
Expanding and Collapsing Vectors, Arrays, Records, or User Defined Buses 4-13
Zooming Waveforms 4-14

Navigating and Viewing the Time Domain 4-15
Measuring Time and Controlling Zoom Regions Using Cursors 4-15
Using Markers to Quickly Display a Time Range at a Later Time 4-16

Finding the Next or Previous Edge on Selected Signals 4-17
Searching a Signal (Vector or Scalar) for a Specified Value . . . 4-17
Expanding Time to Display Exact Order of Events Using Delta Cycles 4-19
Delta Cycle Recording for Verilog. 4-19
Delta Cycle Recording for VHDL 4-20
Expand and Collapse Times 4-20
View Delta Cycles 4-21
Cursor Movement in Delta Cycle Regions 4-22
Advanced Features 4-23
Locating the Cause of Signal Transitions with Event Origin. . . . 4-23
Locating Events Using Breakpoint Groups with Enabled Expressions 4-24
Creating Breakpoint Groups. 4-24
Searching with Breakpoint Groups. 4-27
Collecting and Displaying Unique Events 4-27
Changing the Display Format 4-29
Selecting Drawing Mode 4-30
Modifying Waveform Height. 4-31
Adding and Deleting Blank Lines. 4-31
Displaying Strength Based Colors 4-32
Using the Waveform Style Editor 4-33
Additional Features 4-36
Monitoring a Running Simulation Using Update Mode 4-36
Printing Waveforms 4-38
Printing a Time Range 4-39
Distributing a Printout Across Several Pages. 4-39

Controlling Paper Size and Orientation 4-40
Specifying a Printer Name 4-41
Printing to Postscript 4-41
Adjusting the Signal Name Printout 4-42
Toolbar and Menu Reference 4-43
Toolbar 4-43
Context Sensitive Menus 4-46
Signal Group Pane Context Sensitive Menu (CSM) 4-46
Signal Pane Context Sensitive Menu (CSM) 4-48
Menu Bar 4-49
File Menu 4-50
Edit Menu 4-51
Zoom Menu 4-52
Display Menu 4-52
View Menu 4-53

5. Source Window

Introducing the Source Window 5-2
Execution Pane 5-3
Source Text Pane 5-4
Control Pane 5-5
Before Using the Source Window 5-7
Verilog 5-7
VHDL 5-7
Navigate and View a Design 5-8
Edit the Correct Design File 5-9

Debug Behavior Using Value Annotation 5-9
View Signal Values for All Signals 5-9
View Signal Values for One Signal. 5-10
Follow and Control Source Execution Using Stepping and Breakpoints 5-11
Follow Source Execution 5-11
Control Source Execution 5-11
Using Breakpoints 5-11
Set Breakpoints 5-11
Clear Breakpoints 5-12
Run to Breakpoints. 5-12
Using Instance Groups 5-13
Define Markers. 5-16
Capture Line Data 5-17
Debug Behavior by Analyzing Execution Counts 5-17
Toolbar and Menu Reference 5-18
Toolbar 5-18
Source Text Pane Context Sensitive Menu 5-20
Menu Bar 5-22
File Menu 5-22
Edit Menu 5-23
Display Menu 5-24
View Menu 5-25

6. Logic Browser

Introducing the Logic Browser 6-2

Features 6-2
Getting Started 6-4
Before Using the Logic Browser 6-4
Getting Started. 6-4
Graphic Objects and Value Text 6-5
Viewing Graphic Objects 6-5
Viewing and Loading Value Text 6-7
Navigating a Design 6-10
Navigating the Design with Graphic Objects 6-10
Navigating the Design in the Time Domain 6-12
Navigating Using the Connection Dialog 6-13
Using Event Origin 6-14
Using the Connection Dialog 6-14
Navigating from the Connection Dialog 6-15
Selecting a Signal with Expressions 6-16
Expressions Example 6-17
Editing Source from Logic Browser 6-19
Changing the Display 6-19
Toolbar and Menu Reference 6-21
Toolbar 6-21
Context-Sensitive Menus (CSM’s) 6-23
Logic Browser CSM 6-23
Module Instance CSM 6-25
Port Instance CSM 6-26
Selected Net CSM 6-28

Menu Bar 6-29
File Menu 6-29
Edit Menu 6-30
Display Menu 6-30
View Menu 6-31

7. Register Window

Introducing the Register Window. 7-1
Window Areas 7-1
What is a View? 7-2
Which Signals are Loaded? 7-3
Edit Views. 7-4
Undo Command 7-5
Using Event Origin 7-6
Display Unique Events 7-6
Set Time Scale 7-6
Align Objects 7-7
Distribute Objects 7-7
Display File Designators for SIgnals 7-8
Toolbar and Menu Reference 7-9
Toolbar 7-9
Signal Properties CSM 7-12
Menu Bar 7-15
File Menu 7-16
Edit Menu 7-17

Graphics Menu 7-18
Display Menu 7-19
View Menu 7-20

8. Interactive Window

Introducing the Interactive Window 8-2
History Pane 8-2
Command Prompt 8-2
User-Defined Buttons 8-3
Simulator Controls 8-3
Step Control 8-3
Scope Control. 8-4
Time Display. 8-4
Window Status 8-5
Invoking an Interactive Session 8-6
Starting a Session Via the Simulator Invocation Dialog. 8-6
Starting a Session Via the VirSim Command Line 8-8
Controlling the Simulator 8-9
Summary of Simulator Control Commands 8-10
Breakpoint Durations 8-10
Expression Breakpoints 8-11
Control Simulation from the Waveform Window. 8-12
Control Simulation from the Source Window 8-13
Line Breakpoints. 8-14
Control Simulation from the Logic Browser 8-15
Control Simulation from the Register Window 8-15

Controlling Simulation from Buttons 8-16
Default Buttons Files 8-16
Creating User-Defined Buttons 8-17
Defining a Button 8-18
Selection Substitution 8-19
Toolbar and Menu Reference 8-20
Toolbar 8-20
Menu Bar 8-21
File Commands 8-21
Edit Commands 8-21
Sim Commands 8-22
View Menu 8-23

9. Project Window

Introducing the Project Window 9-1
Defining Projects and Workspaces 9-3
What is a Workspace? 9-3
What is a Project? 9-4
What is a Design Top? 9-4
Workspace Pane 9-5
View Pane 9-9
Workspace Folder View 9-9
Project Folder View 9-10
Design Top Folder View 9-11
Source File Folder View 9-12
Dependency Folder View 9-14

Library Folder View 9-15
Library View 9-18
Tear Off Windows 9-20
Output Pane 9-20
Toolbar and Menu Reference 9-22
Toolbar 9-22
Menu Bar 9-23
File Menu 9-23
Edit Menu 9-24
Project Menu 9-25
Tools Menu 9-27
View Menu 9-28
Windows Menu 9-28
Dialogs 9-29
New Workspace Dialog 9-29
Select a Directory 9-29
Open File Dialog 9-31
Add Files Dialog 9-33
Select a Directory Dialog 9-35
Edit Setup Constants Dialog 9-36
New Verilog Library Dialog 9-38
Library Name 9-38
Path Name 9-38
Ok 9-39
Add 9-39
Cancel 9-39
Help 9-39

Using the Options Dialog 9-39
Options Dialog Buttons. 9-40
Setting Analyze Options 9-41
Setting Elaborate Options 9-42
Setting Simulate Options 9-44
Setting Verilog Compile Options. 9-45
Setting Verilog Run Options 9-46
Executing Commands from the Command Line 9-47
Commands 9-48
Files and Directories 9-54
Moving Projects 9-55
Using a Makefile 9-56

10. Time Units

Display Unit 10-2
Display Precision (Button opens option menu) 10-3

11. Radices

Using a Radix 11-1
Radix Dialog 11-3
Radix Editor 11-4
Mapping List Editor 11-5
Creating a User-Defined Radix 11-6

12. Event Origin

Before Using Event Origin 12-2

Where You Can Use Event Origin 12-2

How to Use Event Origin 12-3

Display the Event Origin 12-5

Choose from Multiple Drivers for an Event 12-6

Event Origin Classifications 12-7

Debug with Event Origin 12-8

13. Building Buses

Using Bus Builder 13-1

Bus Builder Dialog 13-3

Using Buses in the VirSim Windows 13-6

Logic Browser 13-6

Source Window 13-6

Hierarchy Browser 13-7

Waveform Window 13-8

14. Markers

How VirSim Uses Markers. 14-1

Markers Dialog 14-3

Opening the Marker Dialog 14-5

Creating a Marker 14-5

Deleting a Marker 14-6
Editing a Marker 14-6
Setting a Marker 14-6

15. Expressions

Expressions Dialog 15-2
Entering Signal Names 15-6
Opening the Expressions Dialog 15-7
Creating Expressions 15-7
Updating Expressions 15-8
Displaying Expressions 15-9
Trigger Types 15-10
Searching with Expressions 15-11
Level Sensitive Searches 15-12
Edge-triggered Searches 15-13

16. VirSim Setup

Caution: Before You Start 16-2
VirSim Platforms 16-2
X Resources 16-3
Windows NT Registry 16-3
Common Settings 16-4
Mouse Button Setting 16-5

Warning if Configuration Is Modified 16-7
Signals Loaded/Registered with Simulator 16-7
User Defined Unique Event Colors 16-8
Waveform Window Settings 16-8
Source Window Settings 16-10
Hierarchy Window Settings 16-11
Logic Browser Settings 16-12
Register Window Settings 16-13
Interactive Window Settings 16-14
Settings for X Resources 16-15
Tool Tips 16-15
X Color System 16-16
Color Model 16-16
Switching Color Maps 16-17

17. VCD+ (vpd) File Generation

Advantages of VCD+ 17-2
System Tasks and Functions 17-3
System Tasks to Generate a VCD+ File 17-3
System Tasks and Functions for MDAs 17-7
Syntax for Specifying MDAs 17-8
Using $vcdplusmemon and $vcdplusmemoff 17-9
Examples 17-10
System Tasks for Capturing Source Statement Execution Data 17-17
How to Capture Verilog Source Statement Execution 17-18
Source Statement System Tasks 17-19

System Tasks for Capturing Delta Cycle Information 17-20
System Tasks for Capturing Unique Event Information 17-21
Simulator Run-Time Options 17-24
VCD+ Methodology 17-27
Advantages of Separating Simulation from Analysis 17-28
Conceptual Example of Using Verilog VCD+ System Tasks . . . 17-28
VCD+ On/Off PLI Rules 17-31
Performance Tips 17-32

18. Translating VCD and VCD+

vcd2vpd Command 18-2
vpd2vcd Command 18-5

19. Viewing OpenVera Assertions

Introducing OpenVera Assertions 19-2
Built-in Test Facilities and Functions 19-2
How Sequences Are Tested 19-3
Viewing OVA Results in VirSim 19-5

20. Waveform Defaults

Waveform Style: Verilog Defaults A-2
Waveform Style: VHDL Defaults A-4
Waveform Style: EPIC Defaults A-6

Index

About this Manual

This manual explains the use of VirSim for running and debugging active simulations and historical records of simulations and where appropriate, provides conceptual information to help you understand the application of VirSim features.

This preface covers the following topics:

- Audience
- Licensing
- Platforms
- Other sources of information
- Conventions

Audience

The VirSim User Guide provides product description, tutorial, and reference information to help you use the VirSim simulation debug environment.

The information in this document serves as a primary reference source and procedural guide for VirSim users. This guide assumes that the user has the following background.

- Working knowledge of the OSF/Motif™.
- Working knowledge of the Verilog™ language and the construction of designs using Verilog; also a working knowledge of Verilog simulators, or a working knowledge of the VHDL™ language and the construction of designs using VHDL; also a working knowledge of VHDL simulators.

Licensing

Licensing is implemented using the GLOBEtrotter Software product, FLEXlm™. Licenses are checked out as each feature is invoked. For example, a VCD+ license is checked out when a simulation is run to create files for the VirSim environment. The VirSim licenses float and are independent of each other.

Platforms

VirSim is available for both UNIX and Windows. For Unix, the X Resources file defines specific settings for system behavior. For Windows, settings and system behavior are controlled through the Register file. For information on VirSim setup, see Chapter 16, VirSim Setup.

Other Sources of Information

For more information about Virsim and other Synopsys products, refer to the following sections.

Related Publications

Use the following manuals with this manual:

VHDL

- VHDL by Doug Perry, MacGraw-Hill, Inc.
- A VHDL Primer by J. Bhasker, Prentice Hall
- An Introduction to VHDL: Hardware Description and Design by Lipsett and Schaeffer, Kluwer Academic Publishers
- Digital Design and Modeling with VHDL and Synthesis by Chang, CS Press
- VHDL for Designers by Sjoholm and Lindh, Prentice Hall
- 1076-1987 Standard VHDL Language Reference Manual, IEEE Standards, Not available from IEEE. Available from Menchini & Associates, contact mench@mench.com

Verilog

Two books are available from **Open Verilog International,** 15466 Los Gatos Blvd., Suite 109-071, Los Gatos, CA 95032 408-353-8899:

- *The Verilog HDL Language Reference Manual*
- *Programming Language Interface (PLI)*
- *The Verilog-XL Reference Manual*

Commercial book available from Automata Publishing Company, 1072 S. Saratoga Sunnyvale Road, Building A107, San Jose, CA 95129 408-255-0705:

- *Digital Design with Verilog HDL* by Eli Sternheim, Rajvir Singh and Yatin Trivedi

Commercial book available from Kluwer Academic Publishers, 101 Phillip Drive, Norwell, MA 02061 617-871-6300:

- *The Verilog Hardware Description Language* by Donald Thomas and Phil Moorby

Motif

There are many commercially available books on Motif.

- *OSF/Motif User's Guide*, Revision 1.2, from the Open Software Foundation, Inc.

X Resources

- *Xlib Programming Manual*

SOLV-IT! Online Help

SOLV-IT! is the Synopsys electronic knowledge base. It contains information about Synopsys and its tools and is updated daily.

Access SOLV-IT! through e-mail or through the World Wide Web (WWW). For more information about SOLV-IT!, send e-mail to

```
solvitfb@synopsys.com
```

or view the Synopsys Web page at

```
http://www.synopsys.com
```

Customer Support

If you have problems, questions, or suggestions, contact the VirSim technical support in one of the following ways:

- Send e-mail to virsimsupport@synopsys.com
- Call 1-800-837-4564 (1-800-VERILOG)

1

VirSim Overview

VirSim is a modular debug system designed to run in an OSF/Motif™ or Windows NT environment. VirSim can simultaneously post process multiple data files (VCD+, EPIC, etc.) or run an interactive session using a supported simulator. You can view the results of simulations and the source files used to generate the simulations in VirSim windows. Debug sessions can be saved in a configuration file, allowing the session to be resumed.

VirSim is by default installed in each copy of VCS and Scirocco simulators.

Operation Modes

VirSim presents simulation history in two user-friendly operation modes:

- Post Simulation Mode
- Interactive Mode

Post Simulation Mode

VirSim de-couples the debug activity from event driven simulation by using VCD+ history files as its input. In VirSim, we call this post simulation. In post simulation, VirSim uses a Verilog or VHDL simulator in batch mode to generate a VCD+ history file containing the simulation results. Then, by invoking VirSim, any number of users can simultaneously analyze simulation results recorded in the VCD+ file.

Post simulation provides the following distinct advantages:

- Frees the simulator for other users during debugging.
- Many designers can simultaneously use the results of a single simulation.
- Debug is easier and faster when debug is performed separately from interactive simulation.
- An equivalent debug environment is maintained independent of simulators.

Interactive Mode

VirSim runs the simulator and displays simulation data in real-time in VirSim debug windows. You can start the simulation and synchronize debug operations from the Interactive Window. Data is saved to a VCD+ file for later processing. VirSim can run the following simulators interactively: VerilogXL or VCS, and Scirocco.

While in interactive mode, VirSim also can analyze any number of history files from previous runs. This allows for easy comparison with previous results.

Advantages of VirSim

VirSim offers significant advantages in debug analysis technology that dramatically improve the overall speed of design analysis:

An intuitive and easy to use GUI

- Drag and drop of data between windows and panes.
- Configuration files to save and reload the setup of your debug environment.

Powerful debug capabilities allow you to quickly find answers

- Access to source provides exact knowledge on design connectivity.
- An Event Origin command to identify what design elements cause each transition.

- A Logic Browser to clearly present design connectivity: structural and behavioral, fanins and fanouts, values and delay information included.
- A Source Window in which to view, execute or debug the complete source.
- Data from different runs or designs can be viewed in the same window.
- Windows can be synchronized to the same or different times.
- Execution can be backward or forward in time.
- View stand-alone compiled sources to debug Verilog source, even before your fist simulation.

Performance and memory usage are optimized to handle large designs

- Data files use binary VCD+ format for compactness and high-performance access. These can be up to 100 times smaller than VCD files and contain more information.
- Memory is used efficiently during simulation and analysis.
- Algorithms are optimized for fast loading of data and screen updates.

Support of multiple languages, platforms, and simulators.

Key Terms and Concepts

The VirSim User Guide uses the following terms and concepts in feature descriptions and procedures.

Click Left, Click Middle, and Click Right	Procedures in this manual use the terms **click left**, **click middle**, and **click right** to refer to the use of one of the mouse buttons to select or operate on an object. Position the cursor on the object and click the appropriate mouse button.
Drag-and-Drop	The manual uses the term **drag-and-drop** in procedures to refer to the drag-and-drop operation. In VirSim, you can drag scopes and signals from one window or dialog and drop them into another window or dialog. For more information see the Drag-and-Drop section later in this chapter.
Context Sensitive Menus	VirSim relies heavily on hidden Context Sensitive Menus (CSM) in windows and panes. Point at an object, hold the right mouse button (RMB) down, point to one of the choices and release the button. Objects can be signals, instances, ports, nets, values, panes, register memories and transitions. The choices in the CSM are relevant to these objects.
Tear-Off Menus	Any menu with a dotted line at the top can be torn off and placed on the display for easy access. Click middle above the dotted line, and then drag the menu to a convenient position.
Designator	A **designator** is a two character abbreviation for a VCD+. VCD, or EPIC file name, It is frequently used in names to designate the history file for the signal. Compiled source use a Z1 designator. Interactive sessions use an I1 designator.
EPIC	EPIC™ refers to EPIC Design Technology, Inc. and the associated EPIC analog simulators (TimeMill and PowerMill), or the output file generated from those simulators.

Environment OSF/Motif™ and Windows NT™

You can run VirSim in either the **OSF/Motif™** environment or the Microsoft **Windows NT™** operating system. Most screen examples in this manual show VirSim screens for Windows NT. Where features and screen format differ significantly, the manual also includes the OSF/Motif™ format. This manual assumes you have a working knowledge of OSF/Motif or Windows NT.

Scope (Verilog)

A **scope** is any instance of a module, task, function, or named block in the Verilog Hardware Description Language (HDL) source code. These structures are used to organize and describe logical, structural, and functional groupings of code. Modules can contain more modules, tasks, functions, and named blocks. Tasks, functions, and named blocks can only contain named blocks.

Scope (VHDL)

In VHDL, a **scope** is any instance of an entity/architecture component. (The entity/architecture component is analogous to a module in the Verilog language.) Blocks, Packages, Processes, Procedures, and Functions are also considered scopes.

Variables and Signals (Verilog)

The term variables is used interchangeably with the term signals. Variable was adapted from the standard VCD technology and can refer to any of the following Verilog terms:
Net
Reg
Real Number
Integer
Named Event
Time Variable

Variables and Signals (VHDL)

The VHDL language distinguishes between **signals** and **variables**. In general, VirSim uses the term signals to refer to both signals and variables. There are some exceptions to this rule, but these exceptions are either explicitly stated or implied by the context of its usage. (For example, a dialog that has both a selection for Signals and a selection for Variables is obviously distinguishing between the two.) It is also important to recognize that because most variables are dynamic, simulators may not trace them and consequently they will not show up in the hierarchy.

VCD+

VCD+ is a proprietary binary format for simulation history data, which is generated by a PLI program linked to the simulator. It is used for both Verilog and VHDL simulation

Verilog Language	VirSim uses standard Verilog terms including the language, keywords, syntax, system tasks, and PLI calls.
VHDL Language	VirSim uses standard VHDL terms including the language, keywords, syntax, etc.
Window	VirSim uses **windows** to display data and organize features for specific VirSim applications. For example, the Source Window displays source code and provides features for debugging source code.
Pane	**Panes** are sub-windows within VirSim windows that display specific types of information. For example, the Waveform Window contains separate panes for signal names, signal values, and signal waveforms.
View	A **view** is a graphical display in the Register Window consisting of graphics, labels, and values at a specific simulation time.
Group	A **group** is an ordered set of signals that are identified by a group name. Groups are used in the Waveform Window.

VirSim Windows

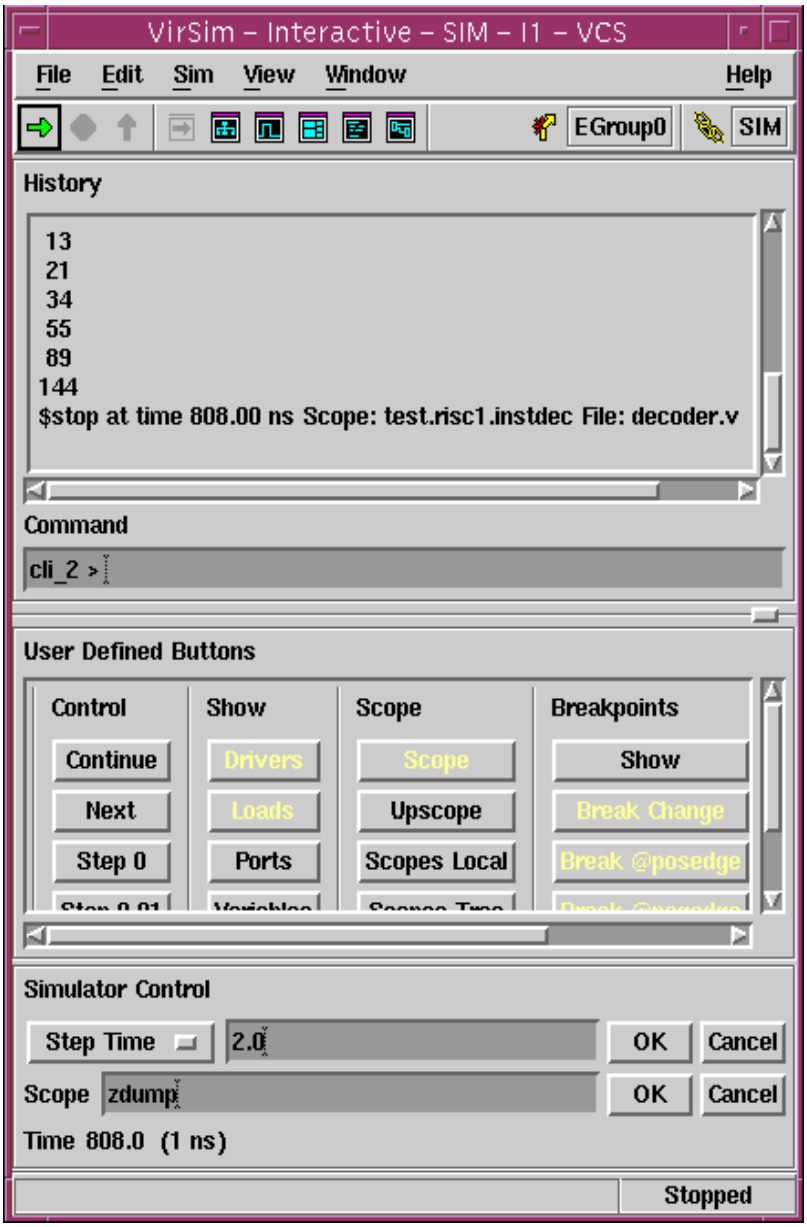
VirSim is a multi-window system. This allows you to have as many of each type of debug window open as desired. Windows can be opened from within any window or with accelerator keys. The following VirSim screens are covered in this section:

- Interactive Window
- Hierarchy Browser
- Waveform Window
- Register Window
- Source Window
- Logic Browser (Verilog only)
- Project Window (VHDL or Mixed VHDL/Verilog)

Interactive Window

The Interactive Window is used to control a simulator and display real time information from the simulator (see Figure 1-1, Sample Interactive Window).

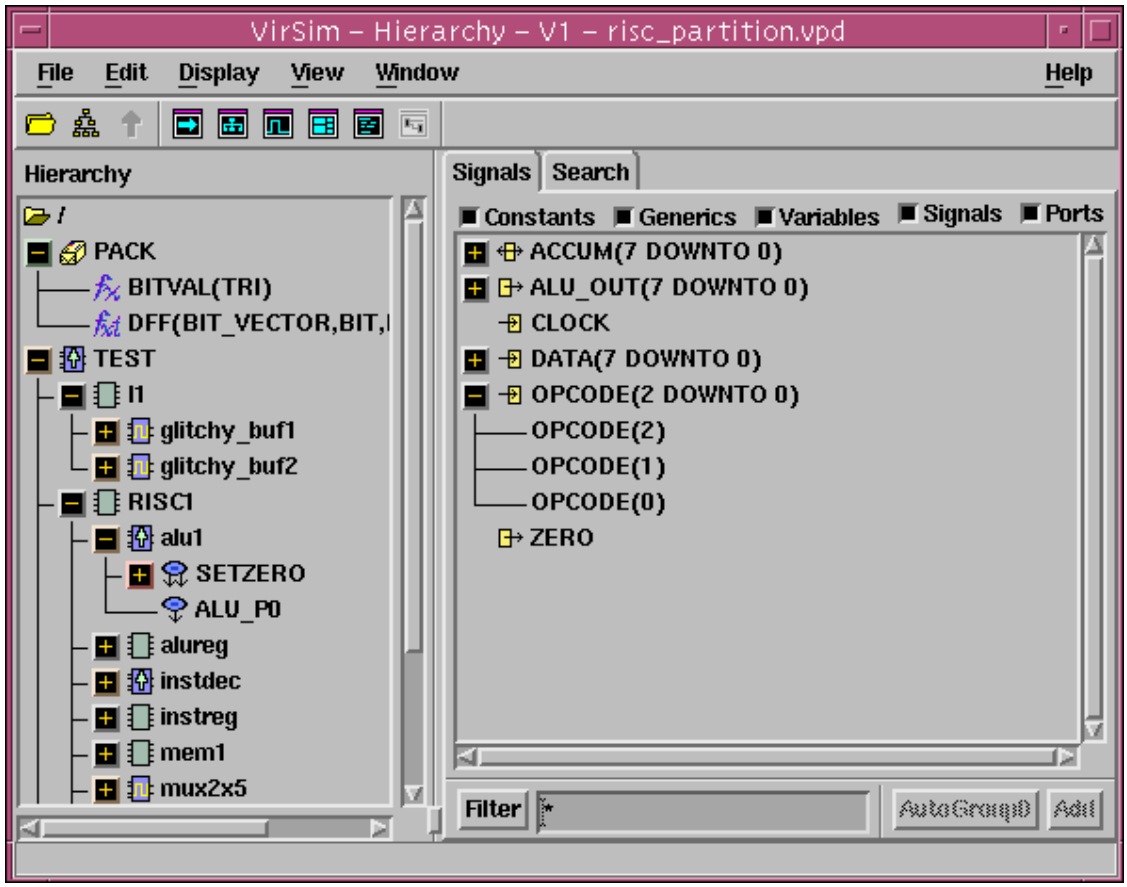
Figure 1-1 Sample Interactive Window



Hierarchy Browser

The Hierarchy Browser displays the design as modules that are color coded to distinguish the different types of modules in the design (see Figure 1-2, Sample Hierarchy Browser. Navigate up and down the design by pointing and clicking on modules and defining roots in the hierarchy tree. When you select a scope, the Signal Select pane can filter and list the internal signals for that scope. You can display objects in other windows by dragging either modules or selected signals from the Hierarchy Browser. If you do not use a configuration file, VirSim opens at the Hierarchy Browser.

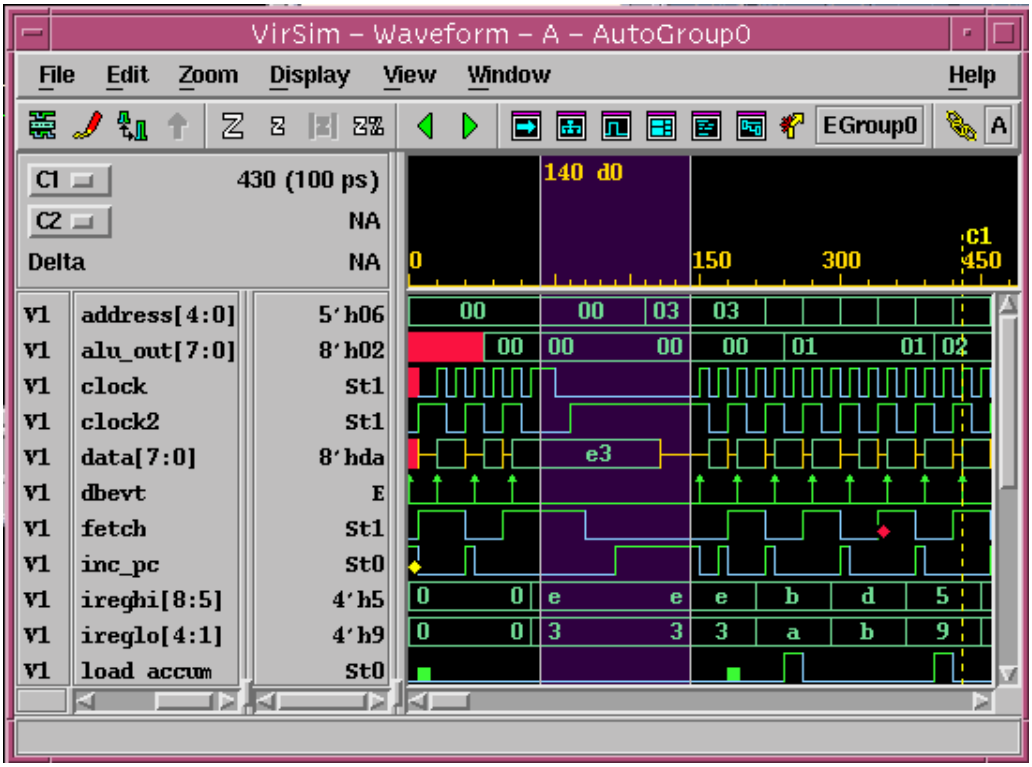
Figure 1-2 Sample Hierarchy Browser



Waveform Window

The Waveform Window displays waveforms for simulation output. Cursors, markers, and expressions let you designate and trace events in simulation time (see Figure 1-3, Sample Waveform Window). Dialogs let you define expressions, buses, signal groups, and breakpoint groups.

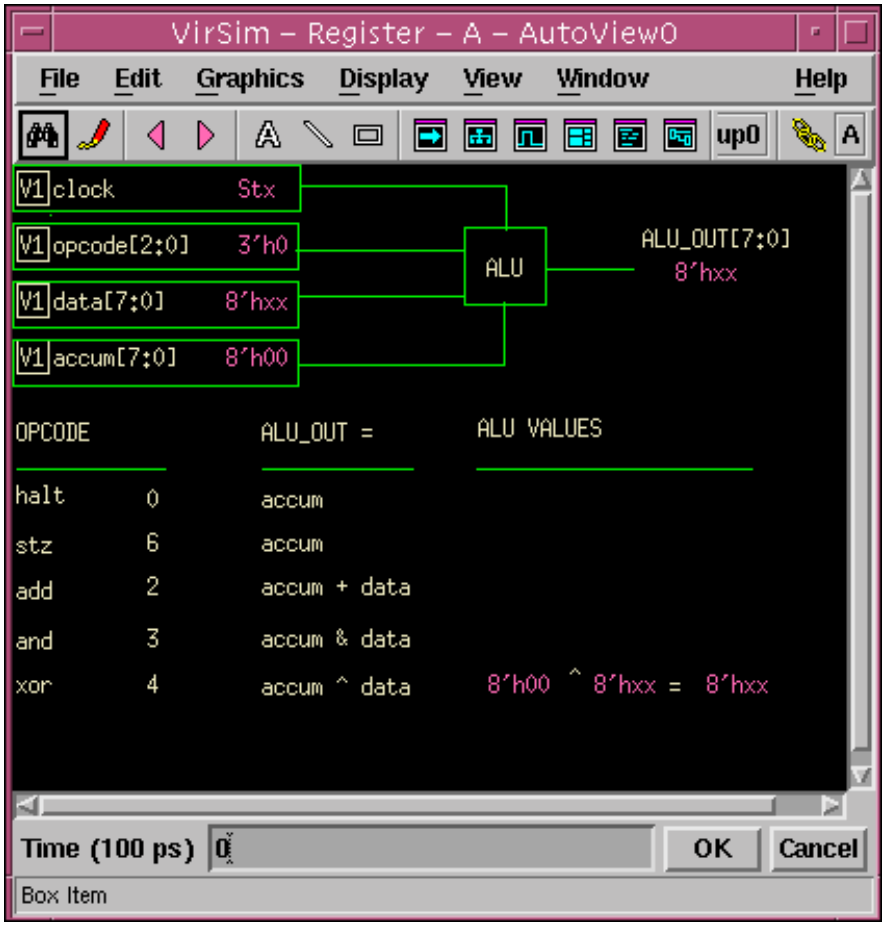
Figure 1-3 Sample Waveform Window



Register Window

The Register Window allows you to create custom views of signals with descriptive text and graphics (see Figure 1-4, Sample Register Window). You can create views by using the drag-and-drop, cut and paste, and editing tools.

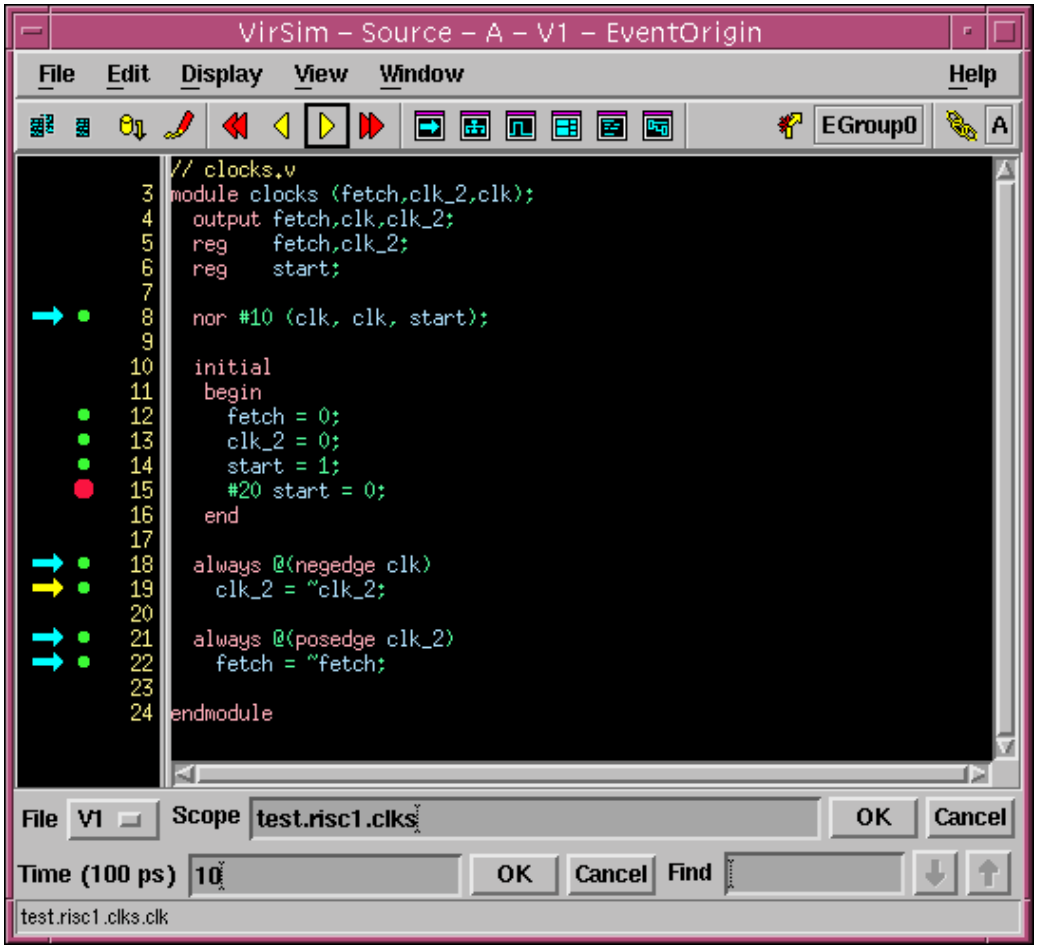
Figure 1-4 Sample Register Window



Source Window

The Source Window displays the code for the selected source instance or groups of instances (see Figure 1-5, Sample Source Window). You may display values of all signals, set and clear breakpoints in this window, step through lines, or run to a breakpoint.

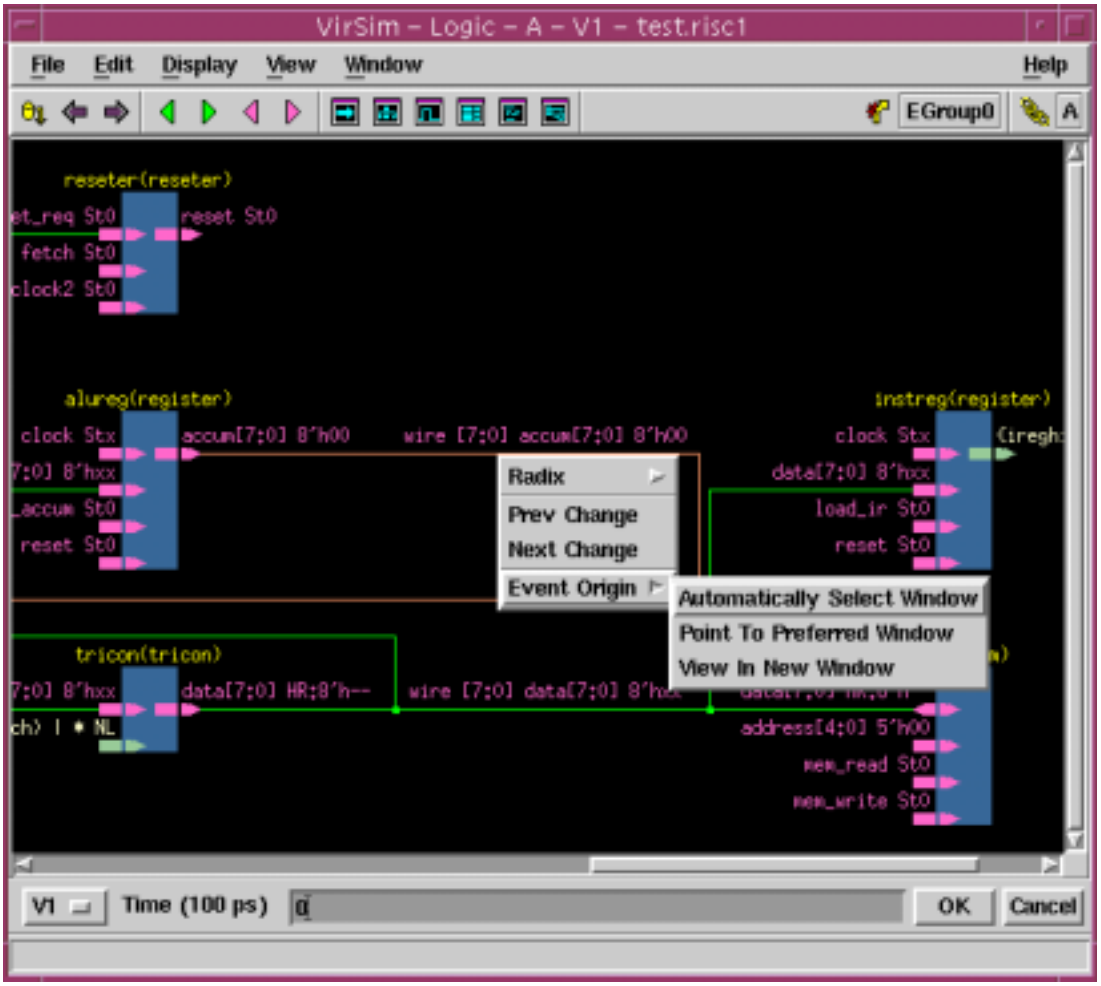
Figure 1-5 Sample Source Window



Logic Browser

(Verilog only) The Logic Browser displays connectivity information (see Figure 1-6, Sample Logic Browser). By selecting a module port or primitive terminal, you can trace net connectivity up and down the hierarchy.

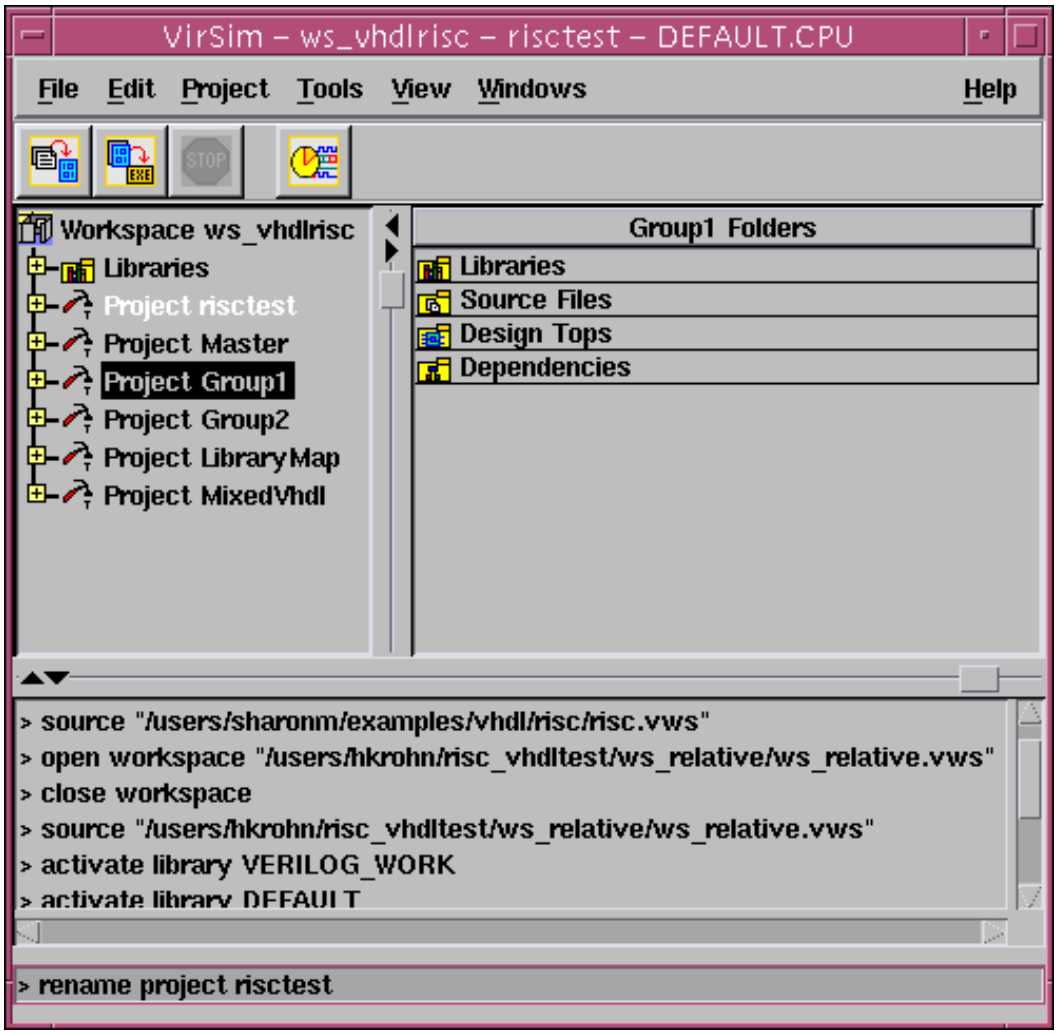
Figure 1-6 Sample Logic Browser



Project Window

The Project Window is used to analyze, elaborate, and simulate VHDL or mixed VHDL and Verilog projects.

Figure 1-7 Sample Project Window



VirSim Navigation

The following section describes features used to navigate in VirSim:

- Using Mouse Buttons
- Selecting Signals
- Drag-and-drop Operation
- Linking Windows
- Accelerator Keys

Using Mouse Buttons

The VirSim system can be configured for either a two-button or three-button mouse. Depending on the type of mouse, the buttons work differently for some operations.

Note:

On some third party X Server packages (such as Exceed) that run on PCs, the two-button mouse can emulate a three-button mouse. In this case, hold down the left and right mouse button on a two-button mouse to emulate the middle button of a three-button mouse.

For maximum efficiency in this highly interactive graphical interface, we strongly recommend the use of a three-button mouse.

Click the left mouse button:

- on a signal or scope to select it.
- on a corner to resize windows and dialogs.

- on a dialog to select menu options.
- in the Waveform pane to set the Cursor 1 (C1) position.
- in the Register Window to select and manipulate graphics tools.

Click the middle mouse button:

- on a signal or scope to drag-and-drop, move, or copy it.
- in the Waveform pane to set the Cursor 2 (C2) position.

Click the right mouse button:

- on a valid object to invoke a context sensitive menu.
- in the timescale area of the Waveform pane to Zoom Cursors.

Selecting Signals

In the Signal Select pane of the Hierarchy Browser and the Signal Name Pane of the Waveform Window, it is often desirable to select a range of signals when dragging or editing. VirSim allows this by using a combination of the left-mouse button and right-mouse buttons with the Shift and Control (Ctrl) keys on the keyboard.

To	Do This	Effect
Select individual signal	Click left on signal	Deselects all previously selected signals. Selects and highlights the new signal. Sets an anchor point at the selected signal for range selection
Deselect all signals	Click left in another window.	Deselects all signals.
Select a sequential range of signals	Either (1) click left on the first signal, press Shift, and click left on the last signal, or (2) click left on the first signal and drag mouse over signals	Selects and highlights all signals in the range.
Toggle select individual signal without deselecting	Press Ctrl and click left on signal	Selects or deselects the signal you are pointing at without deselecting other signals. Resets the anchor point when you want to select/deselect multiple non-contiguous ranges of signals within the same list.

Drag-and-drop Operation

Drag-and-drop is used to move or copy scopes, signals, and text (see Figure 1-8, Drag-and-drop Operation). With a 3-button mouse, hold down the middle mouse button to perform drag-and-drop.

Use the following operations to drag-and-drop objects within the same window (move is the default):

To move objects within the same window

Use drag-and-drop or use Shift+drag-and-drop.

To copy objects within the same window

Use Ctrl+drag-and-drop.

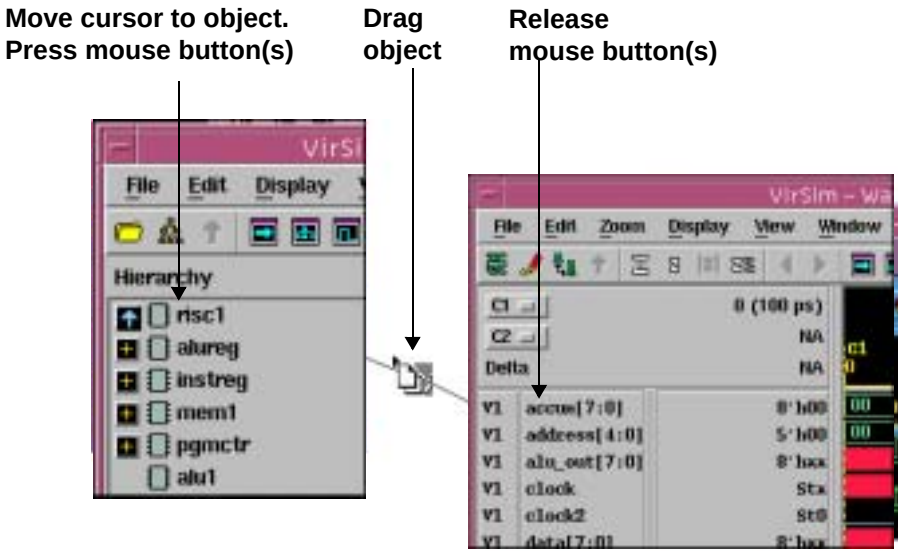
To move objects between windows

Use Shift+drag-and-drop or use drag-and-drop.

To copy objects between windows

Use Ctrl+drag-and-drop.

Figure 1-8 Drag-and-drop Operation



Moving an Array that is Too Large

If during a drag and drop or cut and paste operation the number of elements in an array or radix exceeds the resource value numberOfElementsAllowed default=512, the Sub Range Dialog is automatically opened. The dialog is used to define an object having fewer elements or to cancel the drop operation.

Linking Windows

In linked windows, operations initiated in one window can affect views in other windows. Some VirSim windows (Waveform Window, Source Window, Logic Browser, and Register Window) must be linked in order to synchronize operations, such as breakpoint searches, value changes, and change times. Changing the time in one window automatically changes the view in the other linked windows to that time. The linked windows display a common link letter in the upper-right corner of the window.

There are two ways to link windows:

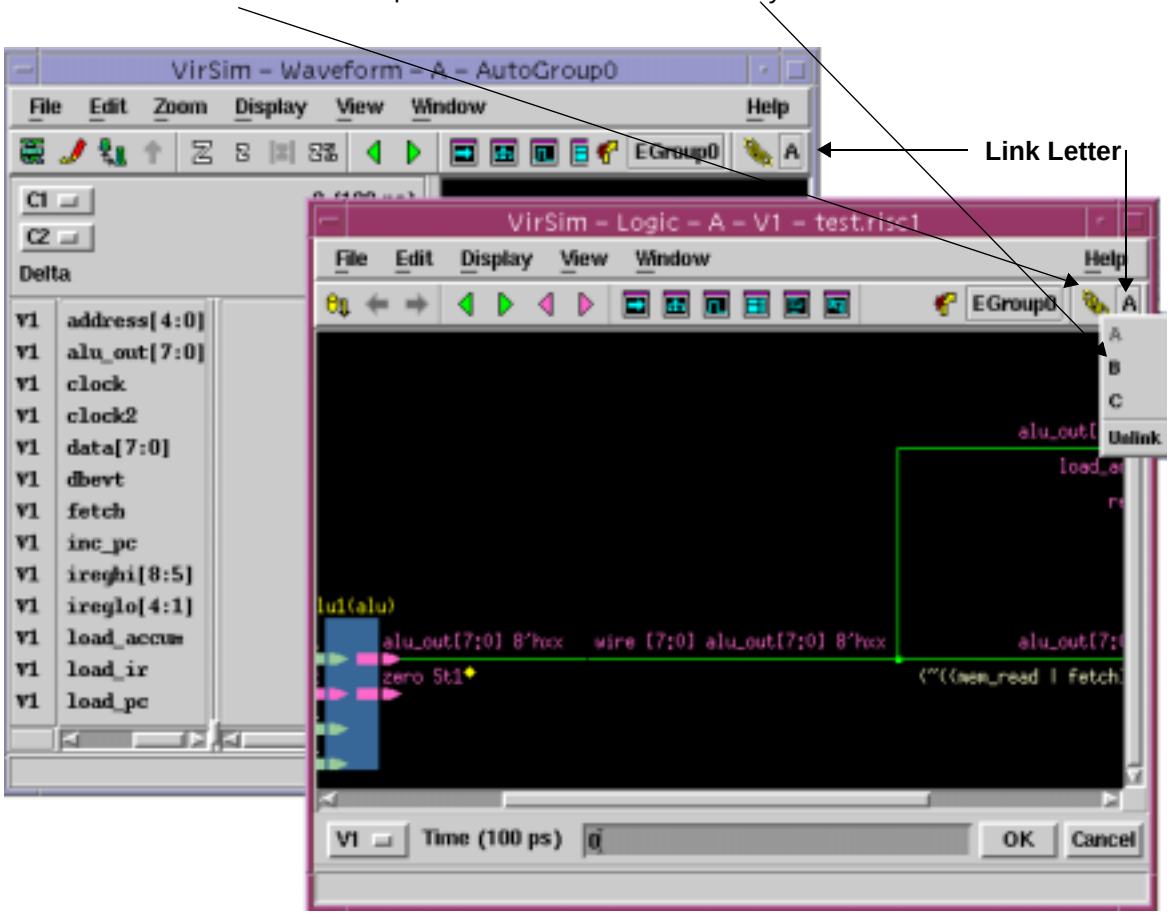
- Open a window from the window to which you want to create a link. Choose the window type from the Window menu.
- For windows that are already open, link the windows with the Link icon (see Figure 1-9, Linking Window).

Figure 1-9 Linking Window

To link windows:

1. Click left on the Link icon to open a link menu.

2. Choose the link letter of the window to which you want to create a link.



Accelerator Keys

By using Accelerator keys, you can perform many menu bar commands quickly from the keyboard. Table 1-1, Common Accelerator Keys, shows commands and accelerator keys for both UNIX and NT Windows environments,and the VirSim windows in which the commands are used. To perform a command, press the accelerator keys indicated in the table.

The Common Accelerator Keys table uses the following codes to identify VirSim windows associated with the accelerator keys.

VirSim Window	Code
Hierarchy Browser	HB
Waveform Window	WW
Source Window	SW
Logic Browser	LB
Register Window	RW
Interactive Window	IW

Table 1-1 Common Accelerator Keys

Operation	Accelerator Keys	HB	WW	SW	LB	RW	IW
Open History File	Ctrl+O	X	X	X	X	X	
Close History File	Ctrl+K	X	X	X	X	X	
Save Configuration File	Ctrl+S	X	X	X	X	X	X
Load Configuration File	Ctrl+L	X	X	X	X	X	X
Print	Ctrl+P		X				
Delete Text in Field	Ctrl+U	X	X	X	X	X	X

Table 1-1 Common Accelerator Keys (Continued)

Operation	Accelerator Keys	HB	WW	SW	LB	RW	IW
Move Cursor to Beginning of Line	Ctrl+B	X	X	X	X	X	X
Group	Ctrl+Shift+G		X				
Views	Ctrl+Shift+V					X	
Marker	Ctrl+Shift+M		X	X	X	X	
Expression	Ctrl+Shift+E		X			X	
Radix	Ctrl+R		X			X	
Zoom In	Ctrl+F		X				
Zoom Out	Ctrl+G		X				
Next Edge	n		X				
Previous Edge	p		X				
New Hierarchy Browser	Ctrl+Shift+H	X	X	X	X	X	X
Reopen	Ctrl+Alt+R	X	X	X	X	X	
Breakpoints	Ctrl+Shift+R		X				
New Waveform Window	Ctrl+Shift+W	X	X	X	X	X	X
New Source Window	Ctrl+Shift+S	X	X	X	X	X	X
New Logic Browser	Ctrl+Shift+L	X	X	X	X	X	X
New Register Window	Ctrl+Shift+R	X	X	X	X	X	X
Edit Parent	Ctrl+Alt+P			X			
Edit Source	Ctrl+Alt+S			X			
Cut	Shift+Del		X			X	X
Copy	Ctrl+Ins	X	X			X	X
Delete	Del		X			X	X
Paste	Shift+Ins		X			X	X

Table 1-1 Common Accelerator Keys (Continued)

Operation	Accelerator Keys	HB	WW	SW	LB	RW	IW
Undo	Ctrl+Z					X	
Stop Simulator	Ctrl+C	X	X	X	X	X	X
Finish Simulation	Ctrl+D	X	X	X	X	X	X
Exit VirSim	Ctrl+X	X	X	X	X	X	X
Invoke Simulator	Ctrl+Shift+I						X
Instance Groups	Ctrl+Shift+I			X			
Clear Instance Breaks	Ctrl+Shift+B			X			
Clear Group Breaks	Ctrl+Shift+G			X			
Load Sources	Ctrl+Shift+A	X	X	X	X	X	X
Move up one signal	Up Arrow		X				
Move down one signal	Down Arrow		X				
Move up one page of signals	Shift+Up Arrow		X				
Move down one page of signals	Shift+Down Arrow		X				
Go to top	Ctrl+Up Arrow		X				
Go to bottom	Ctrl+Down Arrow						
Move right one pane	Right Arrow						
Move left one pane	Left Arrow						
Move right 1/10 of pane	Shift+Right Arrow						
Move left 1/10 of pane	Shift+Left Arrow						
Go to end of time	Ctrl+Right Arrow		X				
Go to beginning of time	Ctrl+Left Arrow		X				

2

Starting VirSim

This chapter describes procedures required to start and run VirSim. There are two ways to start VirSim: entering multiple command line options that specify all the files that you need, or entering only the required options and using the interface to specify the remaining files.

The following information is covered in this section:

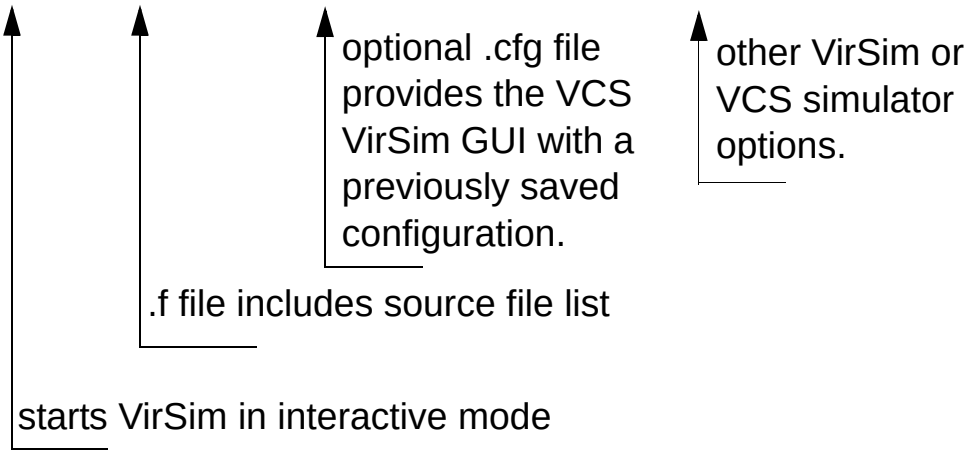
- VCS VirSim GUI Startup Procedures
- Scirocco VirSim GUI Startup Procedures
- Standalone VirSim GUI Startup Procedures
- Opening and Closing History Files
- Saving and Loading a Configuration File
- Loading Sources

VCS VirSim GUI Startup Procedures

Opening VirSim in Interactive Mode

Open the VirSim GUI from the command line using switches to specify how VirSim opens, for example:

```
VCS -RI -f source.f +cfgfile+default.cfg <options>
```


optional .cfg file provides the VCS VirSim GUI with a previously saved configuration.
other VirSim or VCS simulator options.
.f file includes source file list
starts VirSim in interactive mode

`-RI`
Run Interactive. Compiles model for interactive use, invokes the VirSim graphical debugging system immediately after compilation, and pauses simulation at time zero.

`-RIG`
Run Interactive Debug. Like -RI, except you use an existing executable file (such as the simv or simv.exe file) instead of compiling and linking a new one. Even though you have already compiled your source code, you must include the source code on the command line with this option. Also make sure that running VirSim was enabled when you created the simv or simv.exe file, you would have done that with the `-I`, `-RI` or `-PP` compile-time options.

`+cfgfile+<filename>`
(Optional.) Specifies using a configuration file that you recorded in a previous session with VirSim. A configuration file specifies what windows to open and what groups, expressions, etc. to use. You can also use the Load Configuration dialog to specify a configuration file. See Saving and Loading a Configuration File.

`+vslogfile+<filename>`
Enables logging of VirSim commands. If you do not specify a filename, the log is automatically saved to your working directory as VirSim.log.

`+vslogfilesim+<filename>`
Enables logging of simulator communication messages. If you use both +vslogfile and +vslogfilesim, VirSim commands and simulator messages are saved to the same file. If you do not specify a filename, the log is automatically saved to your working directory as VirSim.log.

Creating Value Change Data (VPD) for Post Processing

Modeling the Design

In the Verilog source, save hierarchy and value data for the design by adding the system task: `$vcdpluson(<levels>, <scope>, <signal>)`. See System Tasks and Functions in Chapter 1, VCD+ (vpd) File Generation for additional system tasks.

Compiling the Design

Build the simulator from source files (.v .vhd). The simulator must be built with a VirSim interface so that simulation results can be saved in a compressed VCD+ format. To access the interface, include the -I argument on the command line.

Simulating the Design

Run the simulation using run-time options to modify how VCS writes history files (.vpd). See Simulator Run-Time Options in Chapter 1, VCD+ (vpd) File Generation for a list of run-time switches.

Compile Options for Creating a VPD File from VCS

`-I`

Instructs VCS to automatically include the `+cli` (command line interface), `-P` virsims.tab (default VirSim PLI table), and `-lm` (math library) options. The virsims.tab file defines the VCD+ system tasks, such as $vcdpluson. You need this option, or the `-PP` or `-RI` option, to create VCD+ files. If you use this option instead of the `-RI` option, simulation with VirSim does not start after compilation but you can do postprocessing of the VCD+ file after simulation.

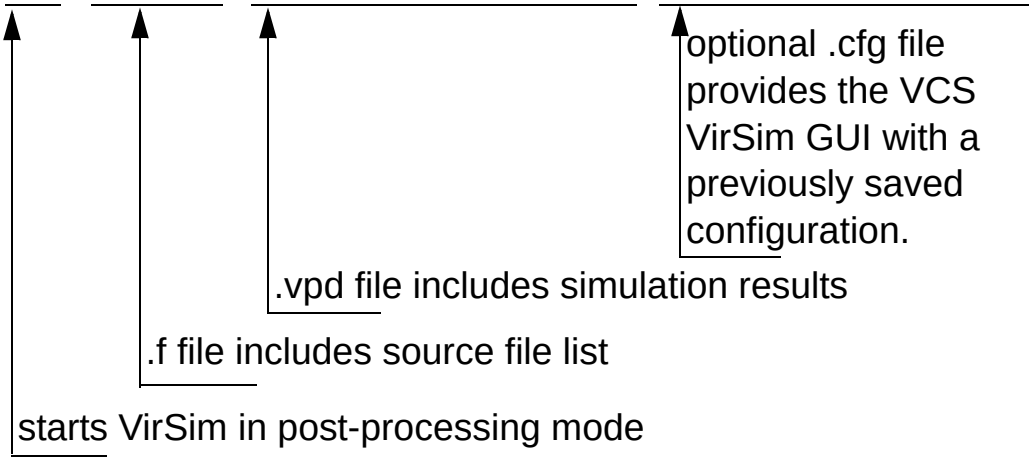
`-PP`

Does the same thing as the `-I` option but minimizes detail of net data for faster post-processing.

Starting VirSim in Post-Processing Mode

Open the VirSim GUI from the command line using VCS compile time options to specify how VirSim opens, for example:

```
VCS -RPP -f run.f +vpdfile+vcdplus.vpd +cfgfile+default.cfg
```


optional .cfg file provides the VCS VirSim GUI with a previously saved configuration.
.vpd file includes simulation results
.f file includes source file list
starts VirSim in post-processing mode

This opens the design in VirSim using a pre-saved environment configuration (.cfg file). For example, your configuration might have opened the Hierarchy Browser, the Waveform Window and the Logic Browser.

If you do not use a .cfg file, VirSim opens at the Hierarchy Browser. From the Hierarchy Browser you can drag and drop signals and scopes into any of VirSim's windows for debug.

Options for Starting VirSim in Post Processing Mode

`-RPP`
Run Post-Processing mode. Starts VirSim for post-process a VCD+ file. Requires a VCD+ file created by the $vcdpluson system task. You can specify the name of the VCD+ file with the +vpdfile option. In post-processing mode VirSim will need to read your source code to use the Logic Browser, so include the source code on the command line with this option.

`+vcdfile+<filename>`
Specifies the VCD file you want to use for post-processing. VCS translates the VCD file to a VCD+ file and loads the new VCD+ file into VirSim.

`+cfgfile+<filename>`
Specifies using a configuration file that you recorded in a previous session with VirSim. A configuration file specifies what windows to open and what groups, expressions, etc. to use.

`+vslogfile+<filename>`
Enables logging of VirSim commands to the specified file. If you do not specify a filename, the log is automatically saved to your home directory as VirSim.log.

`+vpdfile+filename+start+start_time+end+end_time`
For post-processing, specifies the VCD+ file you are using to display simulation results. The optional `+start+start_time` and `+end+end_time` arguments specify you only want VirSim to load and display the results from between these simulation times. For example, to load a `.vpd` simulation file from time 700 to time 1000 without source but with a `.cfg` configuration file in post simulation mode (`-RPP`):

```
vcs -RPP +vpdfile+vcdplus.vpd+start+700+end+1000
+cfgfile+default.cfg
```

Scirocco VirSim GUI Startup Procedures

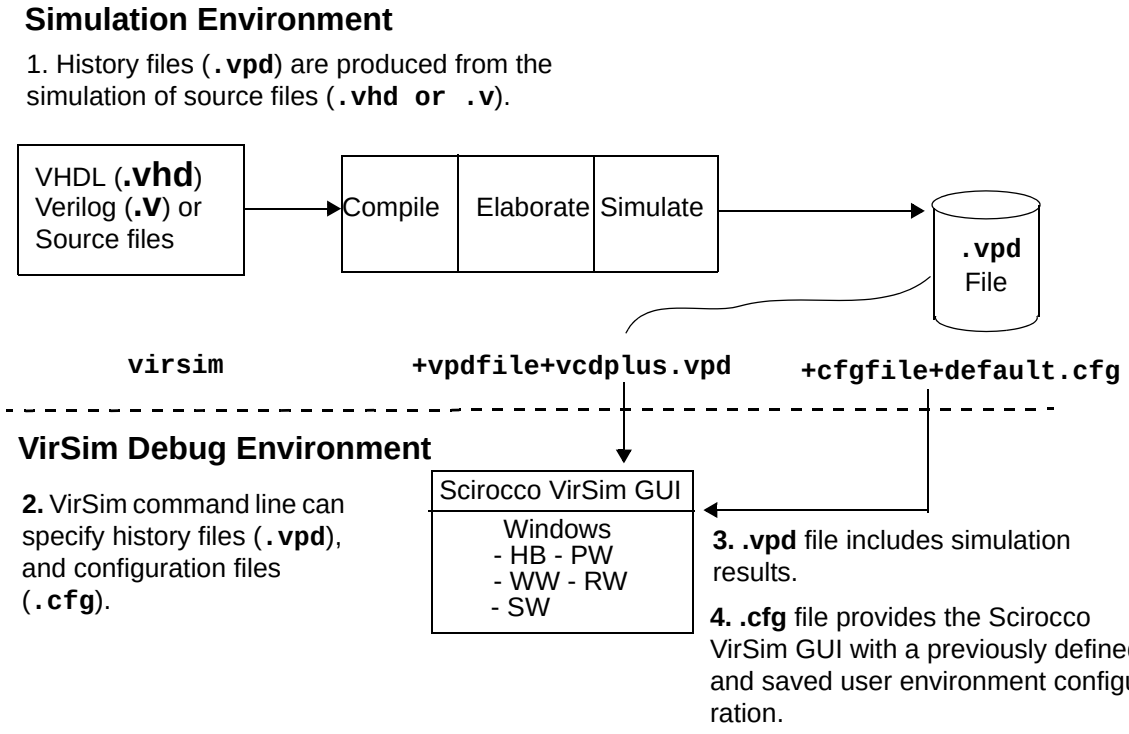
When starting the Scirocco VirSim GUI, the following procedures must be performed in the following order:

1. Compile the design.
1. Elaborate the design.
2. Run the simulation.
3. Start the Scirocco VirSim GUI from the command line.
4. Open history files or start a Scirocco interactive simulation.
5. Load configuration files (optional).

Starting Scirocco VirSim Example

The Scirocco VirSim GUI command line specifies options used to run the Scirocco VirSim GUI. Figure 2-1, Scirocco VirSim GUI Command Line, shows the effect of a command line with arguments in the Scirocco VirSim GUI. Once a simulation has been run, you can load a simulation history file (`.vpd`) in the Scirocco VirSim GUI.

Figure 2-1 Scirocco VirSim GUI Command Line


Simulation Environment
1. History files (.vpd) are produced from the simulation of source files (.vhd or .v).
VHDL (.vhd) Verilog (.V) or Source files
Compile
Elaborate
Simulate
.vpd File
virsim
+vpdfile+vcdplus.vpd
+cfgfile+default.cfg
VirSim Debug Environment
2. VirSim command line can specify history files (.vpd), and configuration files (.cfg).
Scirocco VirSim GUI
Windows
- HB - PW
- WW - RW
- SW
3. .vpd file includes simulation results.
4. .cfg file provides the Scirocco VirSim GUI with a previously defined and saved user environment configuration.

VHDL Command Line Syntax

For VHDL, the Scirocco VirSim command line uses the following syntax:

```
virsim
[+vpdfile+<filename>[+start+<time>+end+<time>]]

[+sim+<simulator-path>] [+simtype+<simulator>]
[+simargs+<parameters>] [+cfgfile+<filename>]
```

Example 1: Load a `.vpd` simulation file from time 700 to time 1000 with a `.cfg` configuration file in post simulation mode.

```
virsim +vpdfile+inter.vpd+start+700+end+1000
+cfgfile+default.cfg
```

Example 2: Start an interactive session and load a `.cfg` configuration file.

```
virsim +simtype+Scirocco +cfgfile+default.cfg
```

Example 3: Start interactive session, load a `.cfg` configuration file, and supply simargs.

```
virsim +simtype+Scirocco +cfgfile+default.cfg
+simargs+"-time ns"
```

Scirocco VirSim Command Line Arguments

The command line arguments are described as follows.

`+vpdfile+<filename>+start+<time>+end+<time>`

The `+vpdfile+<filename>` command automatically loads a VCD+ history file immediately on startup. If Verilog source files are supplied on the command line, it is assumed that these source files match the VCD+ file. You can load multiple files by repeating the `+vpdfile+<filename>` command for each file.

The optional `+start+<time>+end+<time>` command loads a time range from a file. You do not need to specify both times. You can specify just the start load time (to load everything from a certain point on), just an end time (to load everything up to a certain point), both the start and end times (to load everything in a range) or neither (to load all available data from the file)

`+sim+<simulator-path>`

`The +sim+<simulator-path>` command immediately starts an interactive session using the given simulator.

Note:

If the default simulator type is not correct, you must also use the `+simtype` command to select the simulator type.

`+simtype+<simulator>`

The `+simtype+<simulator>` command is used to select the simulator type. When `+simtype` is entered at the command line, the simulator starts automatically and the Interactive Window appears. This argument may be used in conjunction with `+sim` and `+simargs` to override default simulator names and arguments. The Invocation Dialog Type menu lists the legal simulator types.

`+simargs+<parameters>`

The `+simargs+<parameters>` command allows additional arguments to be passed to the simulator when it is invoked. You can specify multiple arguments within double quotes.

`+cfgfile+<filename>`

The `+cfgfile+<filename>` command loads the given configuration file upon startup. A configuration file specifies what windows to open and what groups, expressions, etc. to use.

`+vslogfile+<filename>`

Enables logging of VirSim commands to the specified file. If you do not specify a filename, the log is automatically saved to your home directory as VirSim.log.

Standalone VirSim GUI Startup Procedures

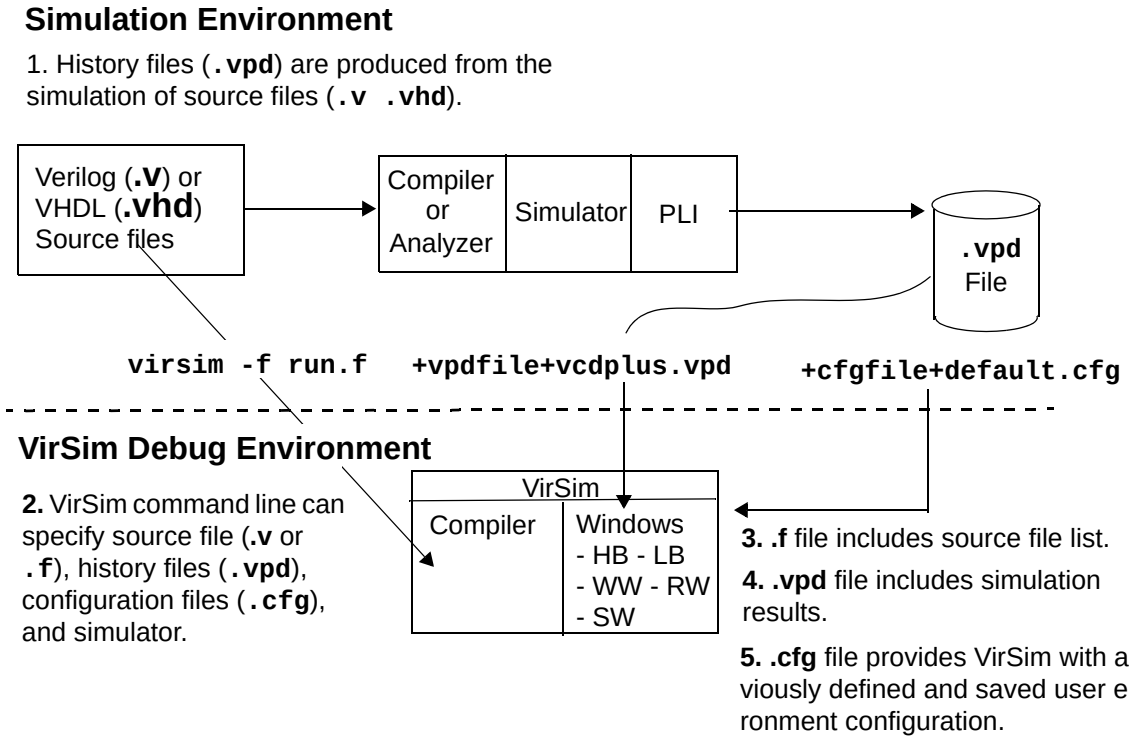
When starting the standalone VirSim GUI, the following procedures must be performed in the following order:

1. Install and setup the environment (see Install Notes).
2. Build the simulator.
3. Run the simulation.
4. Start VirSim from the command line.
5. Open history files or start interactive simulation.
6. Load configuration files (optional).

Starting Standalone VirSim Example

The standalone VirSim command line specifies options used to run VirSim. Figure 2-2, Command Line, shows the effect of a command line with arguments in VirSim. Once a simulation has been run, you can load a simulation history file (`.vpd`) in VirSim.

Figure 2-2 Command Line


Simulation Environment
1. History files (.vpd) are produced from the simulation of source files (.v .vhd).
Verilog (.v) or VHDL (.vhd) Source files
Compiler or Analyzer
Simulator
PLI
.vpd File
virsim -f run.f
+vpdfile+vcdplus.vpd
+cfgfile+default.cfg
VirSim Debug Environment
2. VirSim command line can specify source file (.v or .f), history files (.vpd), configuration files (.cfg), and simulator.
VirSim
Compiler
Windows
- HB - LB
- WW - RW
- SW
3. .f file includes source file list.
4. .vpd file includes simulation results.
5. .cfg file provides VirSim with a previously defined and saved user environment configuration.

Standalone VirSim Command Line Syntax

The command line uses the following syntax

```
virsim [<Verilog Source Info>]
[+vpdfile+<filename>[+start+<time>+end+<time>]]
[+vcdfile+<filename>]

[+sim+<simulator-path>] [+simtype+<simulator>]
[+simargs+<parameters>] [+cfgfile+<filename>]
```

Example 1: Load a `.vpd` simulation file from time 700 to time 1000 without source but with a `.cfg` configuration file in post simulation mode.

```
virsim +vpdfile+vcdplus.vpd+start+700+end+1000
+cfgfile+default.cfg
```

Example 2: Start interactive session and load a `.cfg` configuration file and the verilog source files listed in `run.f`.

```
virsim +sim+/usr/local/simv +cfgfile+default.cfg -f
run.f
```

Standalone VirSim Command Line Arguments

The command line arguments are described as follows.

`<Verilog Source Info>`

The `<Verilog Source Info>` consists of the same command line arguments that are provided to the Verilog simulator. In the previous example, `-f run.f` is used to identify Verilog source (`.v`) files used to run the VirSim simulation. You can also specify source files by listing all the `.v` files at the command line. You can also supply libraries, etc. as specified by standard IEEE-1364 specifications.

In the command line, the source argument is optional. You can also load sources from within VirSim from the Load Sources Dialog described later in this chapter. The Verilog sources are compiled when they are required for use by Virsim, for example when the Logic Browser is used. If you do not enter sources on the command line, the Load Source Dialog opens automatically to request sources when sources are required (see Loading Sources).

`+vcdfile+<filename>`

The `+vcdfile+<filename>` command automatically translates the VCS file to vpd format, then loads it immediately on startup. If Verilog source files are supplied on the command line, it is assumed that these source files match the VCD file. You can load multiple files by repeating the `+vcdfile+<filename>` command for each file.

`+vpdfile+<filename>+start+<time>+end+<time>`

The `+vpdfile+<filename>` command automatically loads a VCD+ history file on startup. If Verilog source files are supplied on the command line, it is assumed that these source files match the VCD+ file. You can load multiple files by repeating the `+vpdfile+<filename>` command for each file.

The optional `+start+<time>+end+<time>` command loads a time range from a file. You do not need to specify both times. You can specify just the start load time (to load everything from a certain point on), just an end time (to load everything up to a certain point), both the start and end times (to load everything in a range) or neither (to load all available data from the file).

`+sim+<simulator-path>`

The `+sim+<simulator-path>` command immediately starts an interactive session using the given simulator. If Verilog sources are supplied on the command line, it is assumed that these source files match the interactive simulation to be started.

Note:

If the default simulator type is not correct, you must also use the`+simtype` command to select the simulator type.

`+simtype+<simulator>`

The `+simtype+<simulator>` command is used to select the simulator type. When `+simtype` is entered at the command line, the simulator starts automatically and the Interactive Window appears. This argument may be used in conjunction with `+sim` and `+simargs` to override default simulator names and arguments. The Invocation Dialog Type menu lists the legal simulator types.

`+simargs+<parameters>`

The `+simargs+<parameters>` command allows additional arguments to be passed to the simulator when it is invoked. You can specify multiple arguments within double quotes.

Note:

For Verilog Users: Because VirSim uses the argument `+vpdfile+` to specify the `.vpd` file to create, and VirSim also uses the argument to open a previously created `.vpd` file, the user may use the `+simargs+` command to pass the `+vpdfile+` runtime argument to the simulator (e.g. `virsim +vpdfile+file1.vpd+simargs+"+vpdfile+file2.vpd"` tells VirSim to open `file1.vpd` for analysis and pass the argument `+vpdfile +file2vpd` to the simulator when it is invoked.

`+cfgfile+<filename>`

The `+cfgfile+<filename>` command loads the given configuration file upon startup. A configuration file specifies what windows to open and what groups, expressions, etc. to use. If you do not use a .cfg file, VirSim opens at the Hierarchy Browser. From the Hierarchy Browser you can drag and drop signals and scopes into any of VirSim's windows for debug.

`+vslogfile+<filename>`
Enables logging of VirSim commands to the specified file. If you do not specify a filename, the log is automatically saved to your working directory as VirSim.log.

Opening and Closing History Files

VirSim provides dialogs for opening and closing history files. You must open a history file or start an interactive session before you can load a configuration. The Open File Dialog and the Close File Dialog are described in the following sections:

- File Types and File Designators
- Opening the Open File Dialog
- Open File Dialog Display Area Descriptions
- Opening a Single History File
- Opening Multiple History Files

File Types and File Designators

The definition of specific file types and file designators follow:

File Types: VCD, VCD+, EPIC, or compiled source (compiled sources are designated with a Z1 identifier). VirSim translates a vcd file to VCD+ before running Virsim.

File Designator: A two-character identifier that differentiates between open files in VirSim. The file designator allows you to differentiate signals with the same hierarchical name but from different simulation history files.

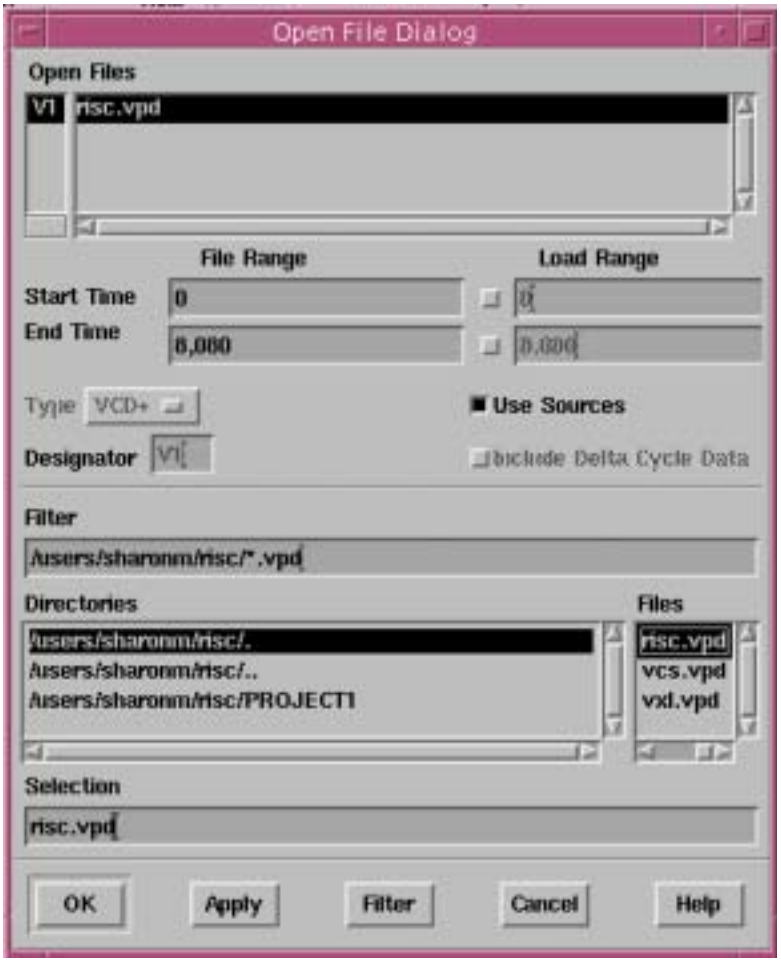
Opening the Open File Dialog

The Open File Dialog is used to open and re-open history files. To open this dialog, choose Open from the File menu in the menu bar or press the Ctrl+O accelerator keys.

Open File Dialog Display Area Descriptions

The Open File Dialog has the following display areas and control buttons (see Figure 2-3, Open File Dialog).

Figure 2-3 Open File Dialog



Open Files

Displays a list of files that are currently open (includes the two-character file designator). If necessary, use the vertical scroll bar to view all files, or use the horizontal scroll bar to read the full filename.

StartTime/End Time

Allows loading a time range from within a file. The File Range fields list the file's time range. The Load Range fields allow you to specify the range that you want to load. Select the button next to the fields that you want to specify.

Type

Allows selection of the type of file to open from an option menu.

Note:

When you open a VCD file, it is first translated to a VCD+ file, which then is opened.

Designator

A unique two character designator for the history file that is used throughout VirSim. The file designator allows you to differentiate signals with the same hierarchical name but from different simulation history files.

Use Sources

(Verilog only) When enabled, Use Sources allows VirSim to use Verilog sources for VCD+ history files that are opened in post simulation. Use Sources is enabled in order to associate loaded Verilog source data with the file that is opened. Sources may be required by the Source Window or Logic Browser. Users may want to disable Use Sources if the sources loaded do not apply to the history file opened.

Use Sources is enabled by default for VCD+ files. Use Sources is not sensitive for EPIC files because the VirSim EPIC interface does not allow sources. For VHDL VCD+ files, the setting of the Use Sources button is ignored.

If the Source Window or Logic Browser requires sources, the Load Sources Dialog will open automatically to request sources. You must then load the appropriate sources in order to continue.

Include Delta Cycle Data (VCD translation only)

Indicates whether or not to filter out delta cycle information during translation of `.vcd` to `.vpd`.

Filter (Motif Only)

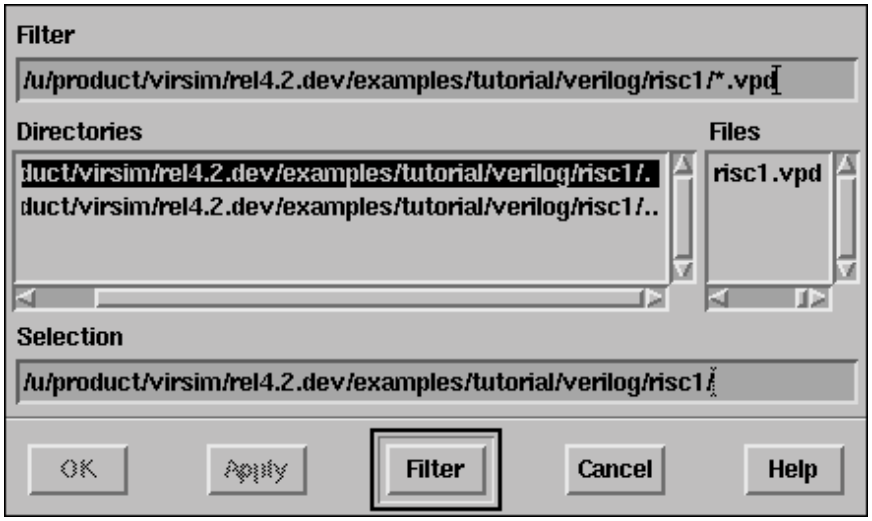
Applies the filter specification entered in the Filter field. The pathname and filename are used in the Filter field to filter the directories and text files. Directories and files that match the filter specification are displayed in the Directories and Files lists.

To display (or update) the files for the selected directory, enter the file path and filter specification in the Filter field and click left on the Filter button. The filter file specification can include the wild card characters * and ?. For example:

```
/u/product/virsim/rel4.2.dev/examples/tutorial/verilog/risc1/*.vpd
```

Figure 2-4, Open File Dialog, Filter Example, shows a filter specification that lists all files with a specific extension, in this case, files with the extension `.vpd`.

Figure 2-4 Open File Dialog, Filter Example



Directories

Initially, shows the current directory (highlighted) and its associated files in the Files list. Click left on a different directory to search for other files or to go up a directory level.

Files

List of files that match the Filter specification. To open a file:

Double-click left on a filename in the Files list, or click left on a filename to select the file, then click left on **Ok** or **Apply**.

Selection

The full filename of the file to open. If a file is selected in the Open Files list or Files list, the filename is automatically shown. If desired, modify this selection to specify the file to open and then click left on **Ok** or **Apply**.

Ok

Opens the file selected in the Files list; this Open File Dialog closes after you click left on **Ok**.

Apply

Opens the file selected in the Files list. The selected file appears in the Open Files list in the Open File Dialog.

Cancel

Closes the Open File Dialog.

Help

Opens user information about the Open File Dialog.

Opening a Single History File

To open only a single history file at a time:

- Select the file and click left on **Ok**.

To reopen a single history file:

- Select the new history file in the Open Files list and click left on **Ok** to open it.

In this case, the previous version of the file is closed, all of the data is deleted, and the new file is loaded. Assuming that none of the signal names have changed, all definitions of windows, groups, and expressions remain unchanged.

Opening Multiple History Files

To open multiple history files:

- Click left on Apply after selecting each file. A unique file designator is automatically assigned to each file.

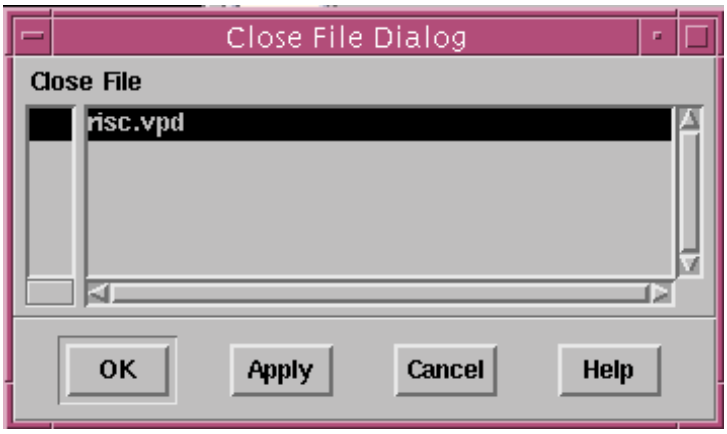
Note:

Once a file is opened, it is not possible to change the Designator. If you are using a configuration file, make sure that the designators created match the designators in the configuration file. The Use Source button is enabled by default.

Closing History Files

The Close File Dialog is used to close history files. To open this dialog, choose Close in the File menu. Figure 2-5, Close File Dialog, shows the file display areas and control buttons for the Close File Dialog.

Figure 2-5 Close File Dialog



Close File

Displays a list of files that are currently open (including the two-character file designator). If desired, use the vertical scroll bar to view all files or use the horizontal scroll bar to read the full filename.

To close a file: Click left on a filename in the Close File list, then click left on **Ok** or on **Apply**. This closes the file, frees all memory used for the file, and removes all signals and scopes from all VirSim windows.

Note:

It is not possible to close an interactive session from the Close File Dialog. Use the Interactive Window to terminate the session.

Caution!

Because closing a file removes all signals from all windows, groups, views, and expressions, you may want to save the current configuration before you close the file.

Ok

Closes the selected file in the Close File list and closes the Close File Dialog.

Apply

Closes the file selected in the Close File list; the Close File Dialog remains displayed after you click **Apply**.

Cancel

Closes the Close File Dialog.

Help

Opens user information specific to the Close File Dialog.

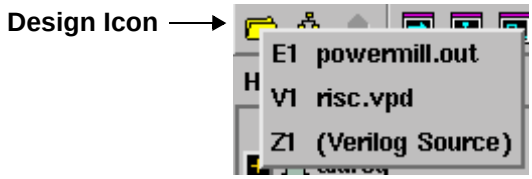
Reopening History Files

The File menu includes a Reopen command that reopens all VCD, VCD+ or EPIC history files that have been changed or updated since the data file was last loaded by VirSim. This command is useful if you debug and resimulate history files but do not exit from VirSim. You can then reopen all files in one operation.

To reopen history files:

1. In the menu bar, click left on File to open the File menu.
2. Select Reopen.
3. To list open files, open the Open File Dialog or click left on the Design icon in the Hierarchy Browser, as shown below. You can choose a file to display in the Hierarchy Browser from this menu.

In the Hierarchy Browser



Saving and Loading a Configuration File

The following section provides guidelines for saving and loading a configuration file. Configuration files are ASCII files that contain information that allows VirSim to start in a predefined configuration. The configuration file can include information, such as open windows, signals, instances, expressions, and breakpoints.

When a configuration file is saved, it includes links to all active design files along with their attributes. This maintains a clear relationship between a configuration file and a set of design files. When you open a configuration file (either from the command line or using the Load Configuration command), you need only specify a configuration file. The designs saved in the configuration file are opened automatically.

VirSim opens design files from the configuration file only when there are no currently opened design files or interactive session and no design file specified on the command line. If you are using multiple configuration files, only the design files found in first configuration file are used.

The following information is covered in this section:

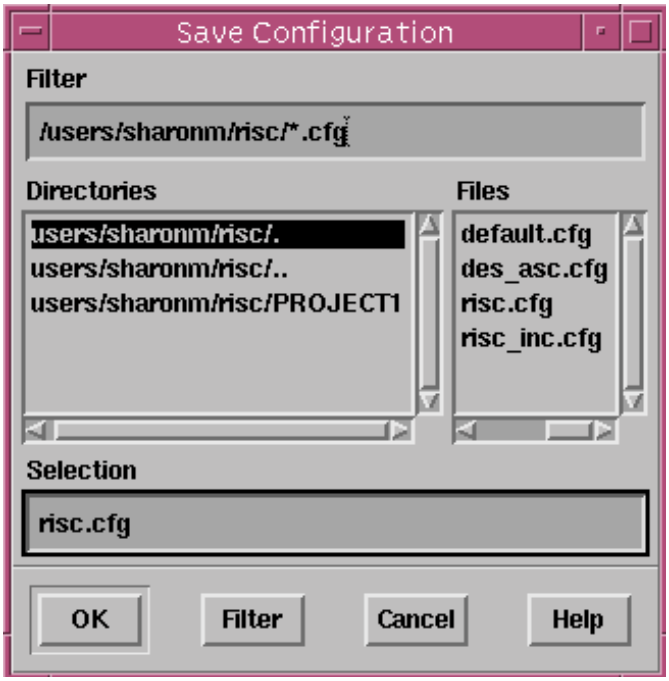
- Saving a Configuration File
- Loading a Configuration File
- Configuration File Format
- Incremental Loading of Configuration Files

Saving a Configuration File

To save a configuration file from a VirSim window:

1. In the window menu bar, click left on File.
2. Choose Save Configuration to open the Save Configuration dialog (see Figure 2-6, Save Configuration).

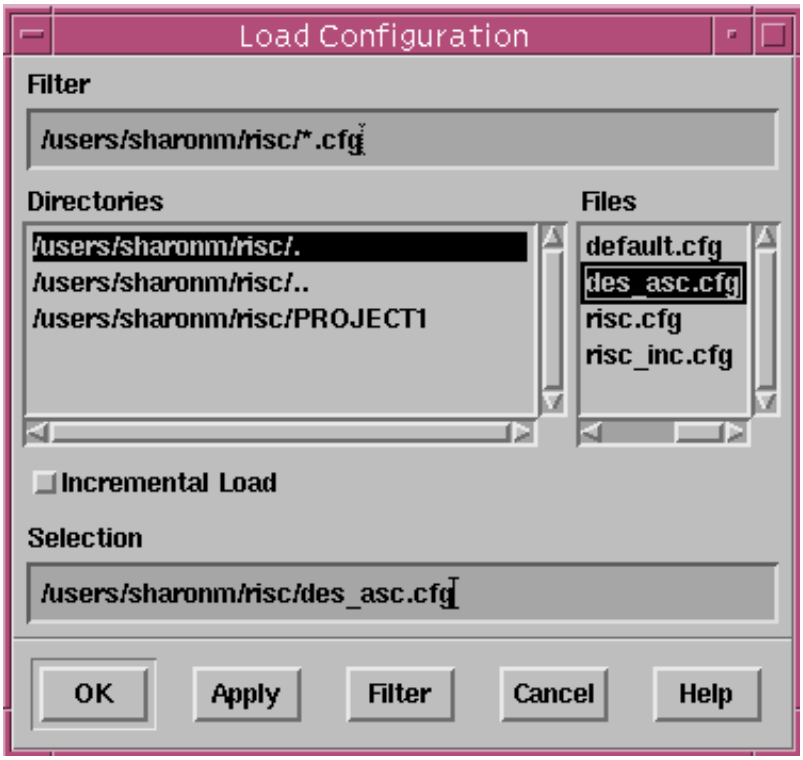
Figure 2-6 Save Configuration



Loading a Configuration File

1. Click left on File in the menu bar and choose Load Configuration to open the Load File Dialog (see Figure 2-7, Load Configuration Dialog).
2. Use the Filter field and Directories field to find the desired file.
3. Click left on the file in the Files field and click left on **Ok**. VirSim restores the configuration defined in the configuration file.

Figure 2-7 Load Configuration Dialog



Configuration File Format

The configuration file can be created or edited with an ASCII text editor independent of VirSim. However, when working with configuration files in an editor, maintain the following order of definitions:

1. Version
2. Design Statements
3. Language Definition (Verilog or VHDL)
4. Buses
5. Expressions
6. Expression Breakpoint Groups
7. Links
8. Markers
9. Radices
10. Register Window Views
11. Waveform Window Groups
12. Source Window Instance Groups
13. Breakpoints
14. Waveform Styles
15. All Open Windows
16. Update Dialog Information

Caution!

The sequence of the definitions in the configuration file must be in the above order. When creating or modifying a configuration file in ASCII, do not alter this order.

Nested Configuration Files

To nest second level configuration files inside a top level configuration file, use an include command as follows:

```
include "~/abc/def/secondlevel.cfg"
```

You can use as many include statements as you like, but all definitions included within both the main configuration file and all included configuration files must follow the order defined in "Configuration File Format" on page 1-29.

Caution!

When you close VirSim, the program asks if you would like to save the current configuration. If you choose to save, VirSim writes over the include statements with the current configuration.

Incremental Loading of Configuration Files

A configuration file can be manually edited and loaded without changing the existing configuration.

1. Click left on File in the menu bar and choose Load Configuration to open the Load File Dialog (see Figure 2-7, Load Configuration Dialog).
2. Use the Filter field and Directories field to find the desired file.
3. Click left on the Incremental Load toggle button.
4. Click left on the file in the Files field.
5. Click left on **Apply**. VirSim incrementally loads the selected configuration file.
6. Repeat steps four and five for any additional configuration files.

Note:

Make sure that the configuration files are self contained. For example, if you define signal groups in the incremental configuration file, everything in the signal group must also exist (i.e., expression, radices, etc.).

Note:

All parts of an incremental configuration file (signal groups, radices, expressions, etc.) must be unique to be loaded. If an element in the incremental configuration file has the same name as a like thing in the existing configuration, that element is not modified.

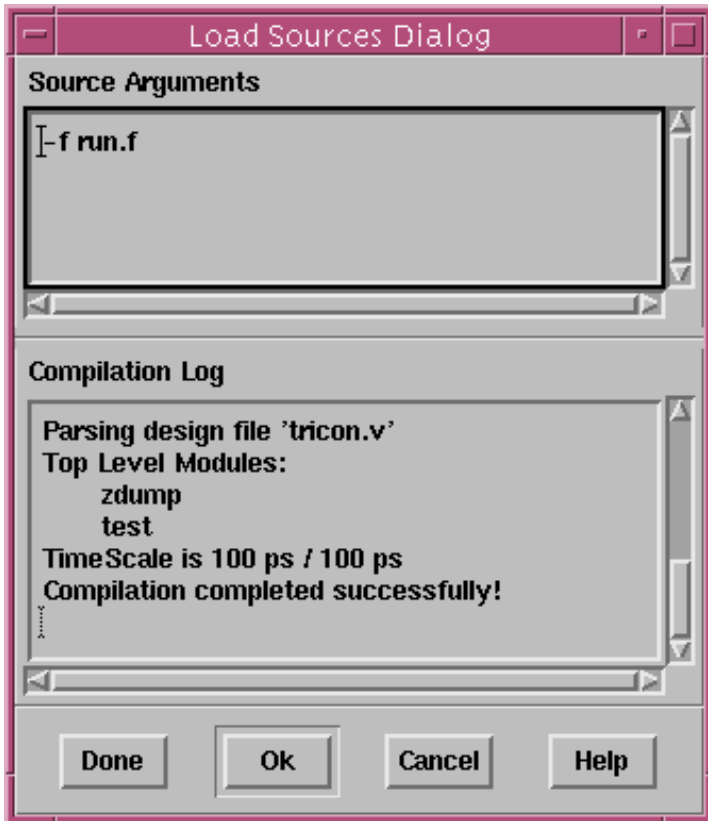
Loading Sources

(Verilog only) The Load Sources Dialog is used to load verilog source files and constructs (see Figure 2-8, Load Sources Dialog). Loaded verilog source files are listed in the open file list with a Z1 designator.

There are two ways to load Verilog sources:

1. The recommended way to load sources is to supply the sources in the VirSim command line. VirSim automatically compiles the sources when they are required for use.
2. If the desired sources are not supplied in the command line, you can load sources in the Load Sources Dialog (see Figure 2-8, Load Sources Dialog). If no sources are loaded in the command line and VirSim requires sources, the Load Sources Dialog opens automatically to request sources. You can also use the Load Sources Dialog to compile sources for stand-alone viewing - no simulation is required and no VCD or VCD+ files need be loaded.

Figure 2-8 Load Sources Dialog



To use the Load Sources Dialog:

1. Choose Load Sources from the File menu to open the dialog.
2. Type the Verilog sources in the Source Arguments field or the name of a file listing all source (as shown above).
3. Click left on **Ok**. The Compilation Log lists the load status of each source as it is compiled. If the sources are successfully compiled, the following message displays:

   ```
   Compilation Completed Successfully.
   ```

   Otherwise, error messages are displayed in the dialog.

Compiled sources are listed in the Select icon list with a Z1 designator. To view stand-alone sources, click the Select Design icon in the Hierarchy Browser and select Z1 Verilog Sources.

4. To close the Load Sources Dialog, click left on **Done**.

3

Hierarchy Browser

The Hierarchy Browser is used to locate and select scopes, signals, and ports (as well as variables, generics and constants for VHDL) for use in other windows. If you do not use a configuration file, VirSim opens at the Hierarchy Browser.

The Hierarchy Browser works on any one of a number of simultaneously open design hierarchies. You may also open multiple Hierarchy Browser windows. By navigating up and down the hierarchy, you can locate the scopes and signals you want, and then drag them to other panes or windows for viewing or signal selection. Filters allow you to selectively locate items for display.

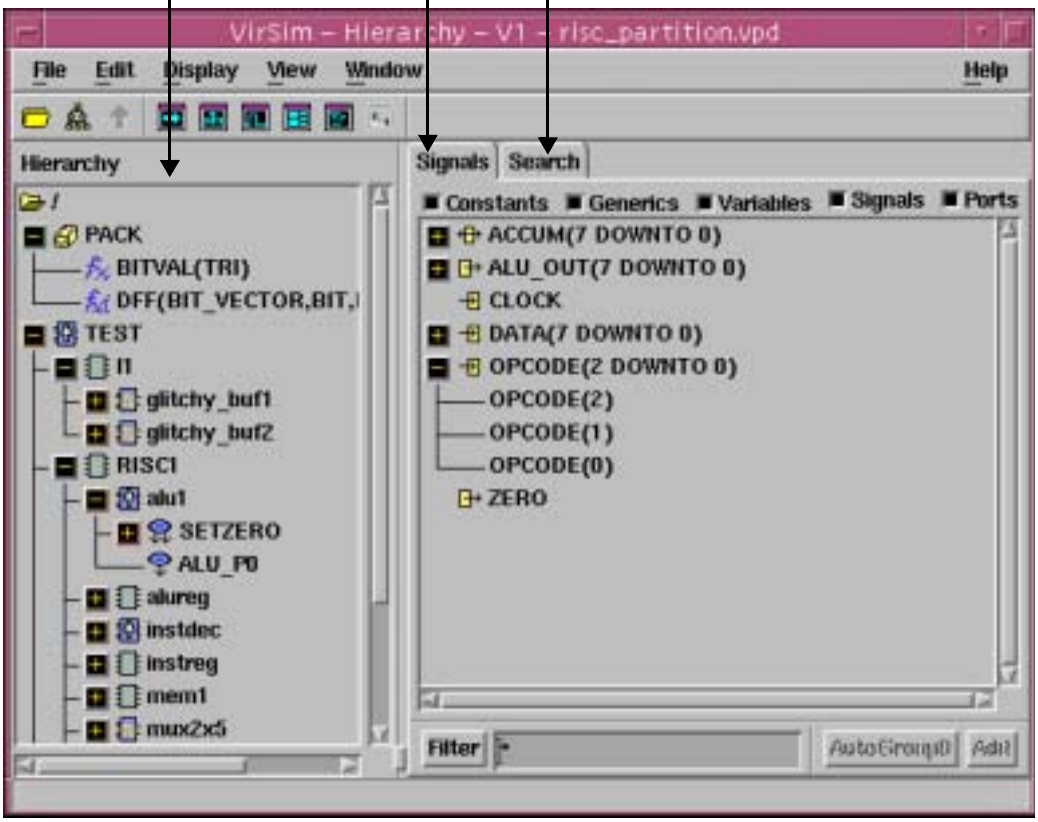
Introducing the Hierarchy Browser

Figure 3-1 Sample Hierarchy Browser

The **Hierarchy Pane** is used to navigate within the design hierarchy. You can expand and collapse the hierarchy, view single or multiple levels, create bookmarks, and drag scopes from this pane to other windows or dialogs.

The **Signals Tab** displays the signals, ports, variables, generics and constants for scopes selected in the Hierarchy Pane. You can drag signals from the Signals tab into other windows or dialogs or add them to signal groups for display in the Waveform Window using the Groups option.

The **Search Tab** is used to locate specific scopes or signals You can drag signals from the Search Tab into other windows or dialogs.



Selecting a Design

To	Do This
Open a history file from the command line.	Specify the VCD file you want to use on the command line when you start VirSim using `+vcdfile+filename`. See Starting VirSim in Chapter 1.
Open a history file from a configuration file.	When you open a configuration file (either from the command line or using the Load Configuration command), if no other design is specified, the designs saved in the configuration file are opened automatically. See "Saving and Loading a Configuration File" on page 1-26.
Open a new history file from the File menu.	1. Click left on the File menu. 2. Select Open. 3. In the Open File Dialog, if necessary, change the directory and file type. 4. Select the History File type (VCD+, VCD, or EPIC). 5. Select the History File.
Reopen history files that have been updated.	1. Click left on the File menu. 2. Select Reopen. All currently open VCD, VCD+, VCD, or EPIC history files that have been changed or updated since VirSim last loaded them are reopened.
Display an open history file or a compiled source file in the Hierarchy Browser.	1. In the Hierarchy Browser toolbar, click left on the Design icon. 2. Select the appropriate history file or compiled source (Z1) file.

Choosing View Options

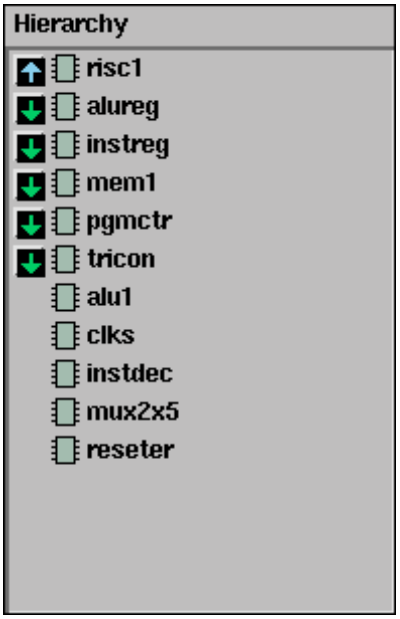
You can view the hierarchy in either Outline or Block Views. For both views, you can choose to view in One-Level or Multi-Level mode. In Block View scopes are identified by labeled buttons. In Outline View scopes are defined by icons. Click the Display menu to select view options. See Figure 3-2, Outline View in One-Level and Multi-Level Modes and Figure 3-3, Block View in One-Level and Multi-Level Modes.

In One-Level View the Hierarchy Pane displays only one level of the hierarchy tree at a time. The parent scope is displayed with an up arrow icon and all child scopes contained within that level are displayed below. The parent scope displays the full hierarchical path and name for its position in the design hierarchy. The child scope names are local to the parent scope

In Multi-Level View the Hierarchy Pane displays all or selected portions of the hierarchy tree. When you select multi-level view, initially only the hierarchy above the currently displayed scope will be added to the display. Once this is done, you can add additional hierarchy to the view by moving up or down the hierarchy, or expanding or collapsing sections of the hierarchy.

Figure 3-2 Outline View in One-Level and Multi-Level Modes

Outline View/One Level



Outline View/Multi Level

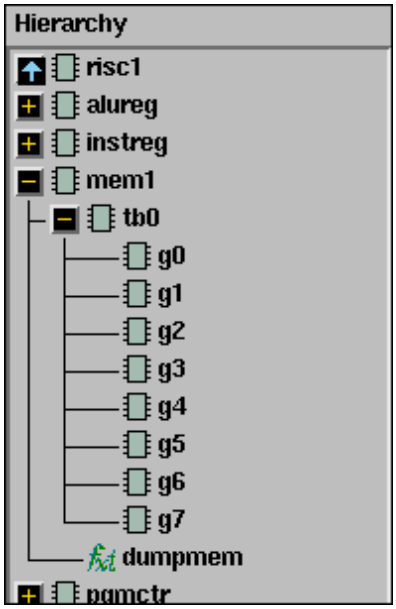
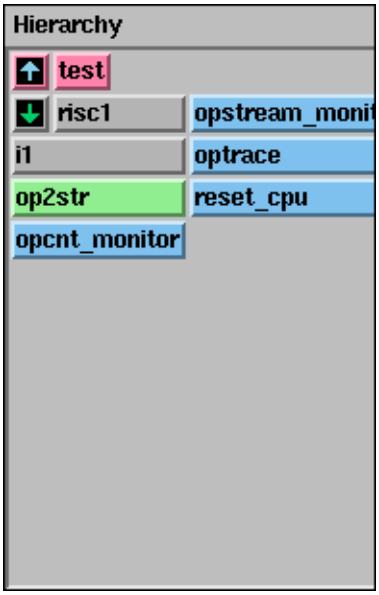
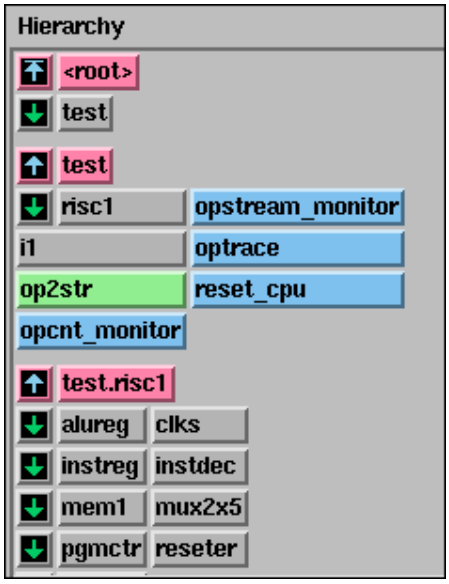


Figure 3-3 Block View in One-Level and Multi-Level Modes

Block View/One Level



Block View/Multi Level



Navigating the Hierarchy

To	Do This
View all top-level scopes in the design at once.	Click left on the Root/Bookmark icon and choose <root>.
Traverse the hierarchy using and arrows. (Arrow hierarchy controls appear in single-level block and outline view and multi-level block view.)	Click left on the up or down arrows in the parent scope. An up arrow indicates additional hierarchy above the scope. A down arrow indicates additional hierarchy below the scope. If a scope does not have any hierarchy below, it does not include a down arrow.
Traverse the hierarchy using and controls. (+/- hierarchy controls appear only in multi-level outline view.)	Click the + and - symbols to expand and collapse the hierarchy tree, or right click to open a CSM and select Expand Tree or Collapse Tree. **Note**: Because the Expand Tree command expands the hierarchy tree for all descendents of the scope, this operation can use a significant amount of memory for large hierarchies.
Bookmark a Hierarchy Pane view.	Click right and Select Create Bookmark.
Jump to a bookmarked Hierarchy Pane view or another root.	1. Click left on the Root/Bookmark icon. 2. Select the desired root or bookmark.
Go to the root scope currently in use.	1. Click right in an unoccupied part of the Hierarchy pane. 2. Select Go To Top.
Drag-and-drop a scope to find its location in the hierarchy.	In any window or dialog, drag-and-drop a scope or signal into the Hierarchy Pane.
Display multiple levels of the design hierarchy in Block View.	1. Click left on the Display menu and choose Multi-Level. At this point the entire hierarchy down to the currently visible instance will be visible. 2. Traverse down the hierarchy using the down arrows. As you move down the hierarchy, additional levels are added to the view.

Searching for Scopes and Signals

The Search Tab is used to find all scopes or signals that match a search pattern and exist in a selected section of the hierarchy. The search results are a function of the currently selected scope, the search toggle buttons (scopes, signals), the search range, and the search pattern. Available ranges are All, Selected, and Children:

Range	Description
All	All scopes or signals in the design hierarchy that match the filter.
Children	All scopes or signals below the selected scope that match the filter.
Selected	All scopes or signals in the selected scopes that match the filter.
Display scopes or signals in the Search Pane.	1. Click left on either the Scopes or Signals toggle button. 2. Click left on the Search button to open the Search menu and select one of the following ranges: All, Selected, or Children. 3. Enter a text string in the Text Entry field to further narrow the range. When used alone, the wildcard (*) displays all signals or ports for the selected range. 4. Click left on a scope to select it.

The filter can accept two wildcard characters: * and ?. The * wildcard can match zero or more characters. When used alone in the filter, it can find all instances in the selected range or scope. The ? wildcard can match any one character.

Filtering Signals Displayed on the Signals Tab

The Signals Tab filter controls can be used to find all signals that match a search pattern and exist in a selected section of the hierarchy.

1. Click left on the Constants, Generics, Variables, Signals, or Ports toggle buttons (only Signals and Ports for Verilog).
2. Enter a text string in the text Entry field to narrow the range. When used alone, the wildcard (*) displays all signals or ports for the selected range.
3. Click left on a signal to select it.

The filter can accept two wildcard characters: * and ?.

- The * wildcard can match zero or more characters. When used alone in the filter, it can find all instances in the selected range or scope.
- the ? wildcard can match any one character.

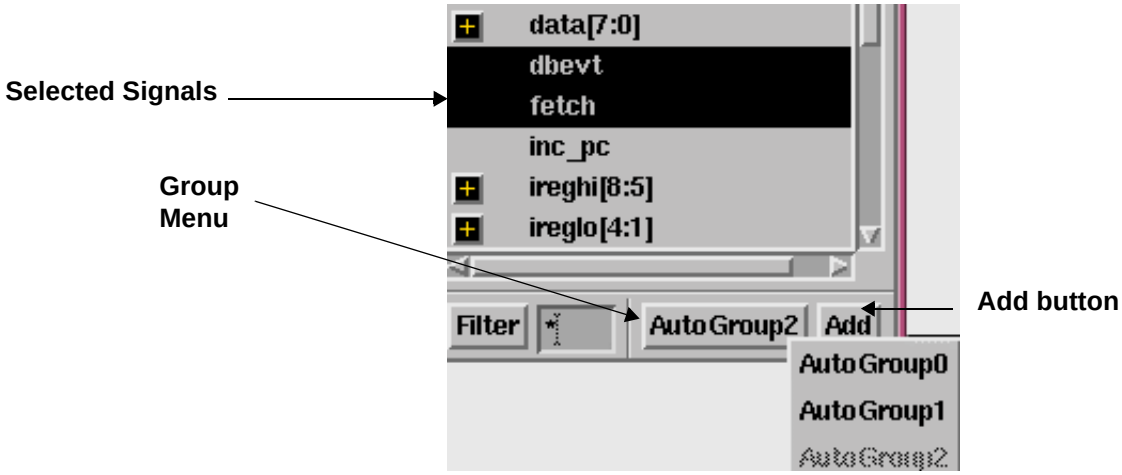
Note:

For simulators that use VCD+ PLI/DKI system tasks, use the +vpdports option on the simulator command line to save port information in a .vpd file. VirSim can then separate signals and ports. If not used, all are considered signals.

Adding Signals to a Group

The Add button adds selected signals to an existing signal group. The Signal Group is selected via the Signal Group menu (see Figure 3-4, Signal Group Menu). Signal groups are used to display related signals in the Waveform Window. When you click left on the Signal Group button, the menu of existing signal groups opens. If there are no available groups, the Group button displays AutoGroup0 (a predefined signal group).

Figure 3-4 Signal Group Menu



To add signals to a group:

1. Select one or more signals in the Signal Select Pane.
2. Click left on the Signal Group button to open the menu and select the group to which you want to add the signals.
3. Click left on Add. The signals are added to the group.

Toolbar and Menu Reference

Toolbar

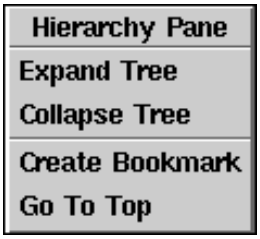
You can toggle display of toolbar icons from the View menu.

Select Design Icon	The Design icon is used to select which design hierarchy to display in the Hierarchy Browser. In post-simulation or interactive mode, the Design icon lists all open history files (VCD+,VCD, or EPIC) and compiled sources. VCD+ and VCD history files are labeled V1, V2, V3, etc. EPIC files are labeled E1, E2, E3, etc. Compiled sources are labeled Z1. In interactive mode, the icon also lists the interactive design, labeled I1. Left click and then drag the cursor to your choice.
Root/Bookmark Icon	The Root/Bookmark icon opens a list of user-defined bookmarks and the <root> or / view. Choose the <root> or / view to display all top-level scopes in the Hierarchy Browser. When you choose a root or bookmark from the menu, the Hierarchy Browser displays the root-level scope or bookmark at the top of the Hierarchy Pane and the portion of the hierarchy tree beginning with the root-level scope.
Update Icon	The Update icon can be used during Update mode to manually read any new design information that has been written to the VCD+ file.
New Window Icons	These icons allow you to open new VirSim windows from the Hierarchy Browser toolbar.

Context Sensitive Menus

Hierarchy Pane Context Sensitive Menu

Figure 3-5 Hierarchy Pane CSM



Expand Tree
Expands the hierarchy tree.

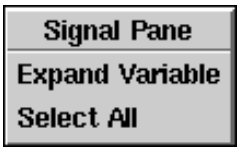
Collapse Tree
Collapse the hierarchy tree.

Create Bookmark
Defines the selected item as a bookmark. To return to the bookmark, click the Root/Bookmark icon and select the bookmark from the displayed list.

Go To Top
Return to the top of the hierarchy.

Signal Pane Context-Sensitive Menu

Figure 3-6 Signal Pane CSM



Expand Variable
Expands the selected variable to display its individual elements.

Select All
Selects all signals in the Signal Pane

Search Pane Context-Sensitive Menu

Figure 3-7 Search Pane CSM

Create Bookmark
Creates a bookmark for a Hierarchy view. The bookmark is added to the Root/Bookmark Icon menu.

Remove Bookmark
Removes a defined bookmark.

Select All
Selects all signals in the Search Pane

Menu Bar

Note:

Any menu with a dotted line at the top can be torn off and placed on the display for easy access. Click middle above the dotted line, and then drag the menu to a convenient position.

File Menu

Open - Opens the file browser to load a VCD+, VCD, or EPIC history file.

Reopen - Reopens all history files that have changed since they were last loaded.

Close - Closes an open history file.

Load Sources - Compiles Verilog source files. Compiled sources are listed, along with history files, when you click the Select Design icon.

Save Configuration - Saves the current configuration.

Load Configuration - Loads a configuration file.

Update - Sets interval for display update.

Close Window - Closes the Hierarchy Browser window.

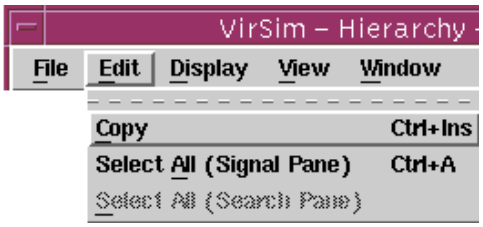
Exit - Exits VirSim.

Edit Menu

Copy - Copies the selection to the Clipboard.

Select All (Signal Pane) - Selects all signals in the Signal Pane

Select All (Search Pane) - Selects all signals or scopes in the Search Pane.



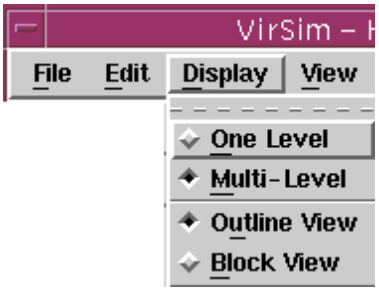
Display Menu

One Level - Displays a One Level view of the scope hierarchy.

Multi-Level - Displays a Multi-level view of the scope hierarchy.

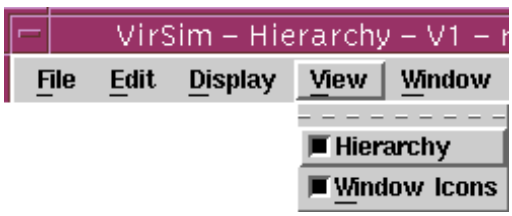
Outline View - Displays an outline view of the hierarchy. See “Choosing View Options” on page 1-4.

Block View - Displays a block view of the hierarchy. See “Choosing View Options” on page 1-4.



View Menu

View Menu settings are saved when you close VirSim.

Hierarchy - Toggles display of the Hierarchy tools (Design, Root/Bookmark, and Update icons).	
Window Icons - Toggles display of the Window icons.	

4

Waveform Window

The Waveform Window is used to analyze signal data over time and to find causes for transitions. Waveform Window features let you select signals to view and navigate the time domain in both simple and complex ways using tools like Delta Cycle, Markers, Event Origin, and Breakpoints.

Introducing the Waveform Window

Refer to Figure 4-1, Sample Waveform Window. The Waveform Window has three major panes:

- The Signal Name Pane displays the names of signals and expressions that are in the currently selected group. By default, the Signal Name Pane only displays the last component of the signal's hierarchical name. To see the signal's full hierarchy name, scroll or enlarge the pane or point at the signal to see the information in the Status Bar.
- The Signal Value Pane displays the signal values in user-selected radices at Cursor 1 (C1) time.
- The Waveform Pane displays the signal waveforms. The time scale at the top of the Waveform Pane shows times at which signals change values. You can expand a sample time to show a Delta Cycle Region which displays signal changes that occurred within a single simulation time (see Expanding Time to Display Exact Order of Events Using Delta Cycles on page 4-19).

These panes may be resized with the pane separator sashes. Each pane can be scrolled horizontally either with X-axis scroll bars or keyboard arrow keys (up and down arrows scroll the pane, shift+arrow moves a page at a time, and Ctrl+arrow moves to the beginning or end of the information in the pane). All three panes are synchronized vertically and can be adjusted vertically with a single Y-axis scroll bar.

In addition, there also are two simpler panes:

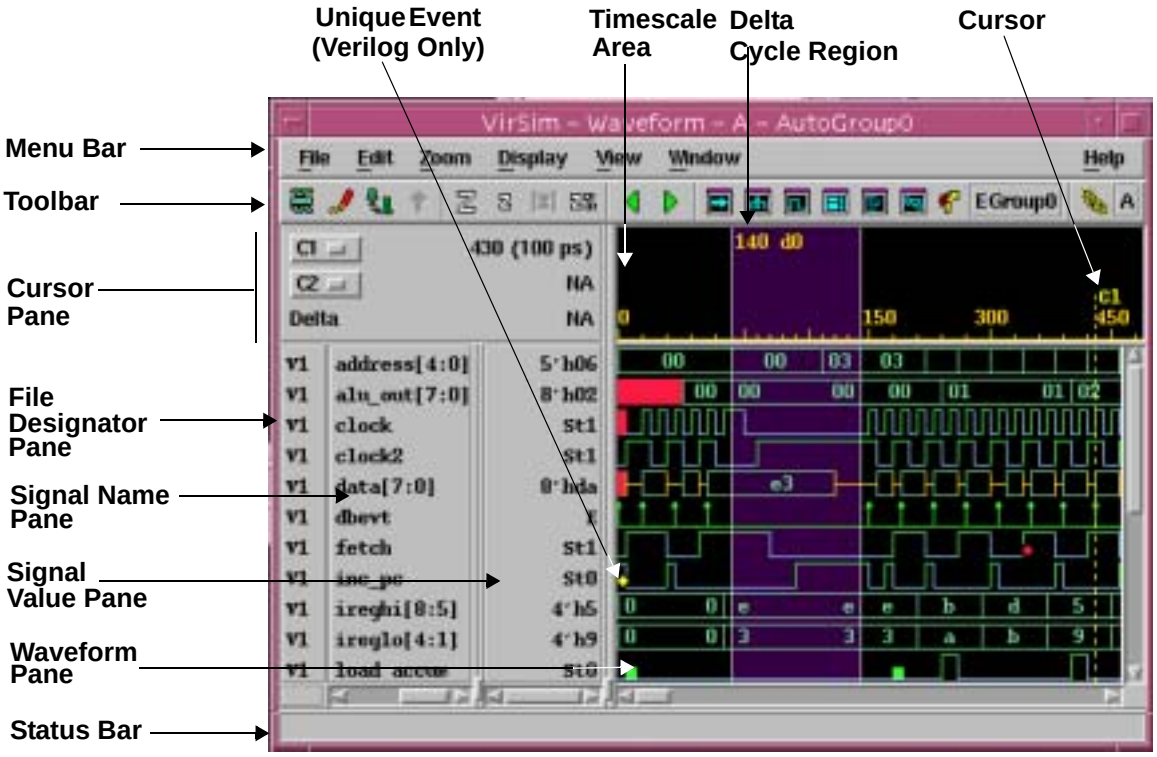
- The Cursor Pane displays additional information about time for the cursors.

- The File Designator Pane displays a two-character identifier associated with each open history file.

The window also has the following standard areas:

- At the top there is a menu bar and a toolbar.
- At the bottom there is a status bar.

Figure 4-1 Sample Waveform Window



Signal Name Pane

The Signal Name Pane displays the names of signals and expressions that are in the currently selected group (see Using Multiple Signal Groups to View Relevant Data on page 4-11). The Signal Name Pane displays the last component of the signal's hierarchical name. To see the signal's full hierarchy name scroll to the left, enlarge the pane, or point at the signal to see the full hierarchical name in the Status Bar.

You can perform the following procedures within the Signal Name Pane:

- Adding and Reordering Signals
- Expanding and Collapsing Vectors, Arrays, Records or User-Defined Buses

Signal Value Pane

The Signal Value Pane displays the value of signals at the time pointed to by cursor C1. Each signal has a default radix, as shown in Table 4-1, Default Radices for Verilog, and Table 4-2, Default Radices for VHDL.

Note:

In Interactive mode, you must unlink VirSim from the simulator in order to display the current value of signals at cursor C1. Otherwise, moving cursor C1 has no effect on the values displayed in the Signal Value Pane, and the value displayed is the last simulated value of the signal (at the Icur cursor).

To change signal radices select a signal or group of signals and right click on the signal value to view the Radix context sensitive menu. Click on a radix enumeration to select it. Select Radix from the Edit menu to create a user-defined radix (see Chapter 11, Radices).

Table 4-1 Default Radices for Verilog

Signal Type	Default	Change Formats
Scalars	Strength	yes
Vectors	Hexadecimal	yes
Integers	2's complement	yes
Time	Hexadecimal	yes
Events	Event	no
Real Numbers	Real	no
Parameters	NA	NA
Expressions	Hexadecimal	yes
Triggers	Event	no

Table 4-2 Default Radices for VHDL

Type of Signal, Variable, Constant or Generic	Default	Change Formats
Bit	BIT	no
Bit Vector	Hexadecimal	yes
Boolean	BOOLEAN	no
Character	CHARACTER	no
String	STRING	no
Enum	determined by enumeration type	no
Integer	signed	yes
Integer vector	INTEGER_VECTOR	no
Physical	first unit	yes

Table 4-2 Default Radices for VHDL (Continued)

Type of Signal, Variable, Constant or Generic	Default	Change Formats
Real	SIGNED	no
Real vector	REAL_VECTOR	no
Record‘	RECORD	no
Std logic	STD_LOGIC	no
Std logic vector	hexadecimal	yes
Time	NS	yes

For information on user-defined radices, see Radix Dialog in Chapter 11, Radices.

Waveform Pane

The Waveform Pane displays the waveforms of the signals and expressions listed in the Signal Name Pane. It also displays the simulation time bar, cursors C1 and C2, markers, names of cursors and markers, and the names of expressions.

This pane also is used to locate an event origin for display in either the Logic Browser or the Source Window. (See Chapter 5, Source Window.)

Also see the following sections:

- Adding and Reordering Signals
- Finding the Next or Previous Edge on Selected Signals
- Monitoring a Running Simulation Using Update Mode
- Measuring Time and Controlling Zoom Regions Using Cursors

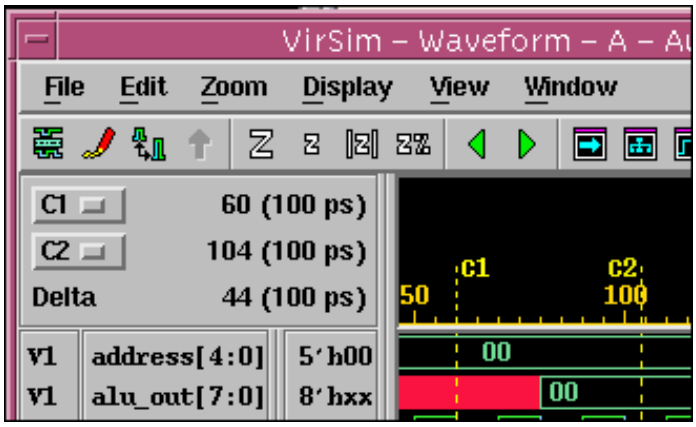
Cursor Pane

The Cursor Pane displays waveform and delta values for cursor types C1, C2, and Icur (Current Time of Simulator). The Icur cursor is displayed only when the Waveform Window is linked to SIM in Interactive mode.

The Delta value displays the difference in time between the two cursor values. To choose the cursor types, select them from the two pull down menus in the Cursor Pane.

Figure 4-2, Displaying Cursor Values, shows an example of cursor values displayed for cursors C1 and C2.

Figure 4-2 Displaying Cursor Values



Status Bar

The status bar, located at the bottom of the Waveform Window, displays the information listed in Table 4-3, Status Bar Information Display, according to the position of the pointer.

Table 4-3 Status Bar Information Display

With Pointer Over:	The Status Bar Displays:
A signal in the Signal Name Pane	File designator, full signal name, value of signal at cursor C1
A value in the Value Pane	File designator, full signal name, value of signal at cursor C1
A file designator in the File Designator Pane	File designator, full filename for the history file
An active toolbar icon button	A brief description of the button function
A digital signal in the Waveform Pane	File designator, full signal name, value of signal at pointer
An analog signal in the Waveform Pane	File designator, full signal name, signal range for the signal, value of signal at pointer (may be interpolated), time

Selecting and Viewing Signals

Adding and Reordering Signals

Adding Signals to a Signal Group

You can add signals to a signal group in one of four ways:

- Drag-and-drop signals from another window to the Waveform Window by dropping them in either of the Signal Name, Signal Value, or Waveform Panes where you want them inserted. When multiple signals are dragged in a single drag operation, they are added to the group in the order in which they were selected.
- Drag-and-drop a scope from another window to the Waveform Window by dropping them in either of the Signal Name, Signal Value, or Waveform Panes where you want them inserted. All signals contained in the scope are sorted in alphanumeric order and then added to the group.
- Select a signal or signals in the Signal Select Pane of the Hierarchy Browser or select a group from the Hierarchy Browser Group icon, then click the Add button. The signals are added to the end of the group.
- Drag-and-drop expressions from the Expressions Dialog. (For information on creating expressions, see Chapter 15, Expressions.)

Reordering Signals in a Signal Group

To reorder signals in the group, perform the following in the Signal Name Pane:

- Drag-and-drop the signal name.
- Cut or copy and then paste using the Waveform Window Edit Menu.

Editing Signal Names and Bit Ranges

You can edit or change a signal name as long as the new name is a valid signal.

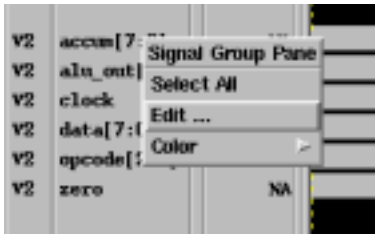
If you select a vector, you can modify the part select as long as the bits you enter are valid. This is useful when you encounter a large vector and just want to view a particular part of it.

To edit a signal or a bit range:

1. In the Signal Name pane, right-click and hold down your mouse over a particular signal name or bit range.

   The Signal Group Pane CSM appears.

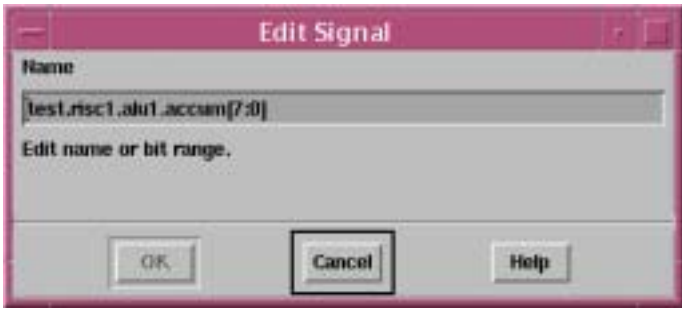
Figure 4-3 Signal Group Pane CSM



2. In the Signal Group Pane CSM, select **Edit...**

   The Edit Signal dialog appears.

Figure 4-4 Edit Signal Dialog



3. Edit the signal name or bit range as appropriate, and click **OK**.

   The signal name and/or bit range of the selected signal changes in the Signal Name pane.

Using Multiple Signal Groups to View Relevant Data

A signal group is an arbitrary set of signals and expressions you group together. Once created, the group is global to all Waveform Windows. You can create an unlimited number of groups and edit any group at any time.

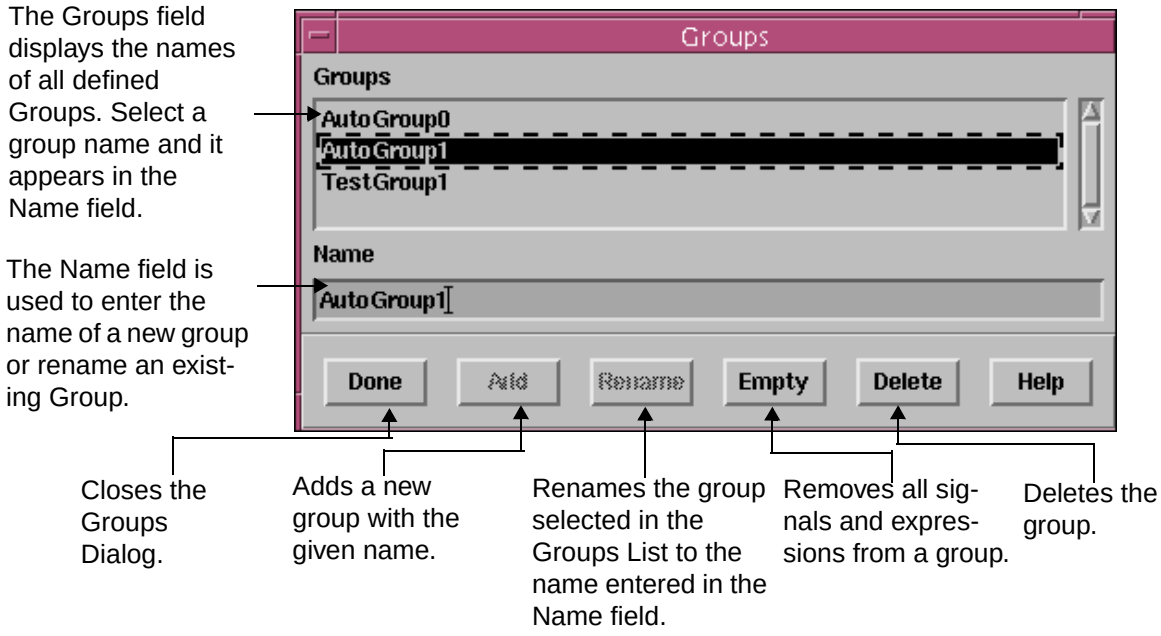
The following information is covered in this section:

- Manually Creating a Signal Group
- Automatically Creating AutoGroups
- Switching Between Multiple Signal Groups

Manually Creating a Signal Group

1. Click left on Edit in the menu bar and choose Groups or select the Group icon and choose New Group... to open the Groups Dialog, See Figure 4-5, Sample Groups Dialog.
2. Click left in the Name field and type a group name.
3. Click left on the Add button, or press Enter, to add the name entered to the Groups list and enable the group for use.
4. Click left on the Done button to close the Groups Dialog.

Figure 4-5 Sample Groups Dialog


The Groups field displays the names of all defined Groups. Select a group name and it appears in the Name field.
The Name field is used to enter the name of a new group or rename an existing Group.
Groups
Groups
AutoGroup0
AutoGroup1
TestGroup1
Name
AutoGroup1
Done
Add
Rename
Empty
Delete
Help
Closes the Groups Dialog.
Adds a new group with the given name.
Renames the group selected in the Groups List to the name entered in the Name field.
Removes all signals and expressions from a group.
Deletes the group.

Automatically Creating AutoGroups

An AutoGroup is a group with a system generated name. If you drag-and-drop a scope or signal into the Signal Name Pane without first selecting a group, VirSim automatically creates an AutoGroup. The name assigned is AutoGroupN where N is a sequentially assigned number starting with 0.

Switching Between Multiple Signal Groups

To switch between multiple signal groups, click the Group icon. This opens a menu containing all defined signal groups. The five most recently used groups appear at the top of the menu, with the most recent group first. The remainder of the menu contains all groups listed in alphabetical order.

Selecting a group loads the signals for that group and displays the group name in the title bar of the Waveform Window. As you move between signal groups, VirSim remembers the most recent position of the vertical scroll bar for each group.

Expanding and Collapsing Vectors, Arrays, Records, or User Defined Buses

Expand an element such as a vector, memory, or record (VHDL) into its components by quickly double clicking its name in the Signal Name Pane. The individual components are displayed directly below the vector or memory. Quickly double clicking again on the expanded element collapses it. Expanded signals can be moved within the window or to another window. Signals moved to another window, or relocated within the same window, remain after the element is collapsed.

Zooming Waveforms

Several tools are available to zoom waveforms:

To select an area to display by dragging the mouse over it.	Click on a waveform. Hold the mouse button down and drag to define a time range. When you release the button, the selected time range will be expanded to fill the horizontal space.
To zoom in.	Click (Zoom in icon). The zoomed display centers around the C1 cursor.
To zoom out.	Click (Zoom out icon). The zoomed display centers around the C1 cursor.
To zoom to a selected percentage.	Click (Zoom Percent icon). Displays a pop-up menu of zoom percentages: 100%, 75%, 50%, and 25%.
To zoom to cursors C1 and C2.	Click (Zoom Cursors icon). This zooms the Waveform Pane to display a time range between cursor C1 and C2. You can also zoom the cursors by right clicking in the Timescale Pane or selecting Zoom Cursors from the context sensitive menu.
To zoom vertically to compress Waveform.	Click (Vertical Compress icon). This toggles between normal and condensed waveforms.

Navigating and Viewing the Time Domain

The following information is covered in this section:

- Measuring Time and Controlling Zoom Regions Using Cursors
- Using Markers to Quickly Display a Time Range at a Later Time
- Finding the Next or Previous Edge on Selected Signals
- Searching a Signal (Vector or Scalar) for a Specified Value
- Expanding Time to Display Exact Order of Events Using Delta Cycles

Measuring Time and Controlling Zoom Regions Using Cursors

Table 4-4, Cursor C-1 and C-2 Positioning, explains how to position cursors C1 and C2 (using default placement). You can move the cursors to the nearest signal edge or to a specific signal time. If there are no edges or you click in the Time Scale Pane area, the cursor moves to the position where you clicked the mouse. The default behavior can be changed in the X resources file. For more information, see Chapter 16, VirSim Setup.

Table 4-4 Cursor C-1 and C-2 Positioning

Two-Button Mouse	Three-Button Mouse	With Pointer on Verilog Signal	With Pointer on Analog Signal	With Pointer on Analog Signal Displayed as Digital	With Pointer in Timescale Pane
Click Left	Click Left	C1 to nearest edge	C1 to time	Snap C1 to 50% of threshold	Sets C1
Shift + Click Left	Shift + Click Left	C1 to time	C1 to time	Snap C1 to 50% of min/max	--
Ctrl + Click Left	Click Middle	C2 to nearest edge	C2 to time	Snap C2 to 50% of threshold	Sets C2
Shift + Ctrl + Click Left	Shift + Click Middle	C2 to time	C2 to time	Snap C2 to 50% of min/max	Sets C2
Click Right	Click Right	CSM	CSM	CSM	Zooms area between C1 and C2

Using Markers to Quickly Display a Time Range at a Later Time

Markers function as user-defined aliases for specific simulation times. They are used to bookmark specific points in simulation time in VirSim windows so that specific simulator times can be easily accessed.

Finding the Next or Previous Edge on Selected Signals

To find the next or previous edge of a signal or group of signals:

1. Select the signal(s).
2. Position the mouse in the Waveform Pane and right click to open the Waveform Pane CSM.
3. Select Next Edge or Previous Edge from the CSM.

   VirSim locates the next or previous edge that occurs in any of the selected signals.

OR

1. Select the signal(s).
2. Use the N and P shortcuts in an active Waveform Window.

Note:
Find Next/Previous Edge also stops on any expression breakpoint that is turned on in the currently active breakpoint group.

Searching a Signal (Vector or Scalar) for a Specified Value

To find the next or previous specified value of a signal or group of signals, using the Expression dialog, create an expression with the signal name and the Verilog expression you want, then enable the search for the expression in the Waveform window.

1. Choose Expressions from the Waveform Window Edit menu. This opens the Expressions dialog.

2. Enter a name for the expression in the Name field.
3. Drag the signal(s) of interest into the Expression pane (or enter the full hierarchical signal name).

Note:
The expression must be legal Verilog code and can consist of multiple signals with associated values.

4. Enter the value that you want to search for. For example, an expression that searches for a value might look like the following:

```
V1$test.risc1.data[7:0] == 8`h91
```

Or an expression that searches for multiple values might look like the following:

```
$V1$test.risc1.alu_out[7:0] == 8'h02 & $V1$test.risc1.pcaddr[4:0] == 5'h06
```

5. Click Add to place the expression in the list of Breakpoints.
6. To enable the Breakpoint for searching in the Waveform Window, click the button next to the breakpoint name. This enables the Search Prior and Search Next expression icons in the Waveform Window.
7. In the Waveform Window, click the Search Next icon to search for the next point at which the expression is true. VirSim places a marker with the expression name at the point where the expression is true.

Note:
You can also drag expressions from the Breakpoint pane to the Waveform and Register windows to quickly find the areas where the expression is true - even for expressions that consist of multiple signals and conditions.

For detailed information on creating and using expressions, refer to Chapter 15, Expressions. Also see Locating Events Using Breakpoint Groups with Enabled Expressions on page 4-24.

Expanding Time to Display Exact Order of Events Using Delta Cycles

The Delta Cycle feature allows you to display detailed value change data, or delta cycles, that occurred within single sample times. If, for example, at time 800 you see that 8 signals changed state, you could expand time 800 to show the sequence of signal changes. Delta Cycle is also useful when investigating glitches or race conditions, where a signal may change multiple times within one sample time.

Delta Cycle Recording for Verilog

Enable reporting of delta cycle information in the VCD+ file using the `$vcdplusdeltacycleon` command to use this function in post-processing mode. For more information, refer to Chapter 17, VCD+ (vpd) File Generation, System Tasks for Capturing Delta Cycle Information.

When running in interactive mode, you can toggle recording of delta cycle information in the Interactive Window by selecting Delta Cycle Recording from the Sim Menu.

Note:

When you select Delta Cycle Recording in Interactive mode, actual recording starts at the beginning of the next time sample. If you are at or past the time for which you want delta cycle information, turn on Record Delta Cycle Data then from the Sim menu select Re-Exec to rerun the simulation.

Delta Cycle Recording for VHDL

You can enable the capture of delta cycle information before simulating your design and not during runtime. To capture delta cycle information, set the VPDDELTACAPTURE variable to ON or OFF (default is OFF) in the .synopsys_vss.setup file or use the command line switch -vpddeltacapture on|off when simulating.

When viewing the simulation results in the waveform window, you can observe changes at the delta cycle level by pointing at a signal transition with the mouse and using the right-mouse button. A popup menu appears, select the Expand time option to open up the series of transitions to show the sequence of delta cycle changes.

The trade-off for capturing delta cycle information is longer simulation runtime and larger VPD databases. Therefore, use this feature only for debugging delta delay problems.

Expand and Collapse Times

To expand a time to view delta cycle information:

1. Right click on the edge of a waveform that includes a delta-cycle event to open the Waveform Pane CSM.
2. Select Expand Time.

You can collapse either all delta cycle regions or a selected delta cycle region. To collapse all delta cycle regions, right click anywhere in the Waveform Pane and select Collapse All Time from the CSM. To collapse a single delta cycle region, right click in the region to open the Waveform Pane CSM and select Collapse Time.

View Delta Cycles

When you expand a time, a delta cycle area is inserted into Waveform and Timescale panes at the time currently pointed at. Refer to Figure 4-6, Sample Delta Cycle Region. Only cycles that have changes on signals active in the window are displayed, and the cycles are evenly spaced. Spacing for delta cycles is controlled with the VirSim*deltaCycleWidth variable in the Resource file, refer to Chapter 16, VirSim Setup. The default is 8 pixels.

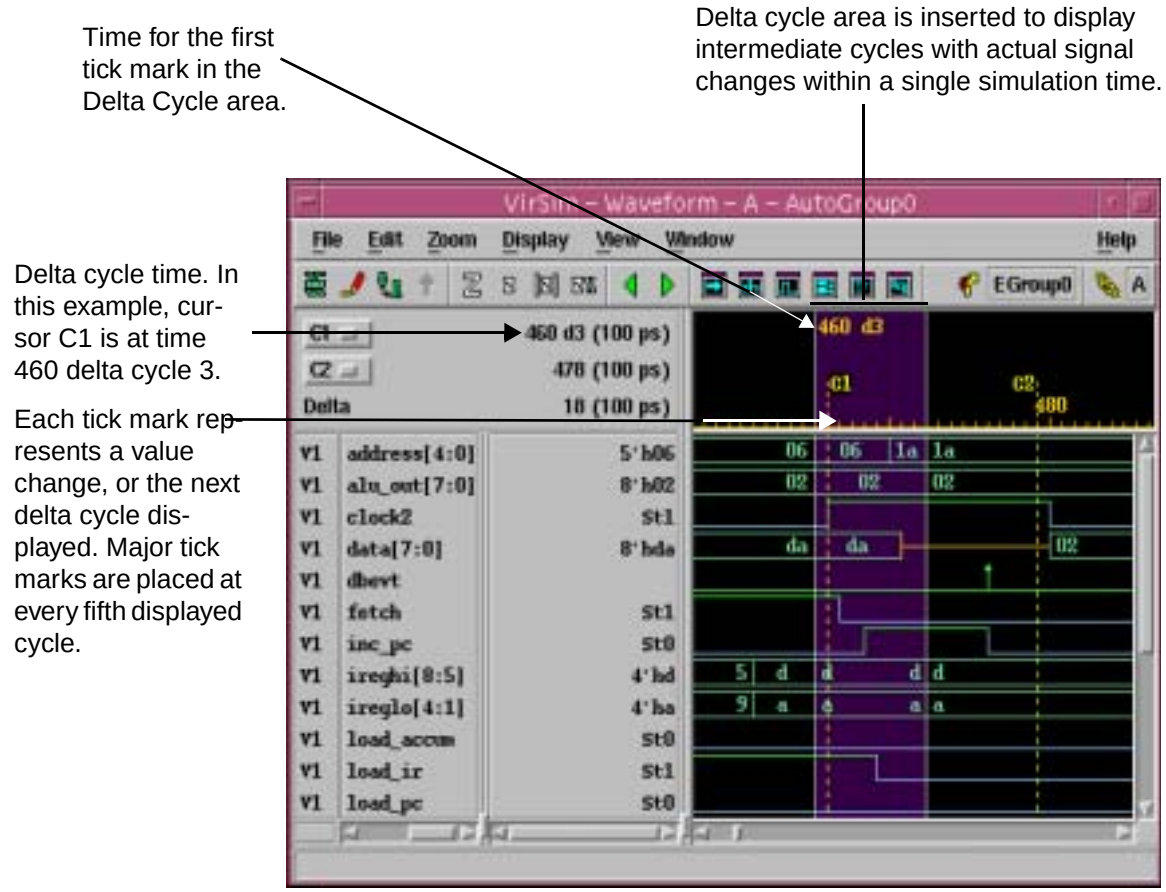
When you change the precision or units of the Waveform Window, all current delta cycle regions are collapsed. If there are any files open with a loss of precision with respect to the current display precision, delta cycle is disabled (insensitive) until the precision changes.

Note:

Delta cycle waveform areas do not zoom.

The group displayed in the Waveform Window might contain a mixture of signals with and without delta cycle information (for example, if some displayed signals are from a file for which delta cycle information was not saved). In this case, any edge for a signal without delta cycle information appears in the last delta column. This column has an asterisk above the tick in the timescale region. If you move the mouse over this column, the status bar gives a time such as 12,350 dNA indicating delta cycle data was not available.

Figure 4-6 Sample Delta Cycle Region



Cursor Movement in Delta Cycle Regions

Click left in the Timescale area to snap cursor C1 to the nearest delta cycle. Click middle in the Timescale area to snap C2 to the nearest delta cycle. When placed within a Delta Cycle region, C1 and C2 times are displayed in the delta cycle time format in the C1 and C2 fields of the cursor information area.

When both cursors are placed in the same delta cycle region, the Delta field (time between C1 and C2) displays the number of delta cycles between cursors. If the cursors are in different delta cycle regions or if only one cursor is in a delta cycle region and the other is in nonexpanded time, the Delta field displays the difference between the two time samples with no reference to delta cycles.

When C1 is at the right edge of a Delta Cycle Area (at the end of a sample time), information is presented as if at the end of the expanded sample time.

Advanced Features

The following information is covered in this section:

- Locating the Cause of Signal Transitions with Event Origin
- Locating Events Using Breakpoint Groups with Enabled Expressions
- Collecting and Displaying Unique Events

Locating the Cause of Signal Transitions with Event Origin

Using the position of the mouse pointer as a reference, the Event Origin feature can obtain precise information about the source of a signal value change at a specific time. Click right on a waveform edge and hold the mouse button to open the Waveform Pane CSM to use the feature. This is a Verilog only feature. For more information, see Chapter 12, Event Origin.

Locating Events Using Breakpoint Groups with Enabled Expressions

A breakpoint group defines a subset of all defined expression breakpoints for which expression search is enabled. An expression is added to a breakpoint group by enabling it for that breakpoint group in the Expressions Dialog. Expressions can be enabled for more than one breakpoint group. When searching with a breakpoint group, VirSim stops when any expression enabled in the breakpoint group is satisfied.

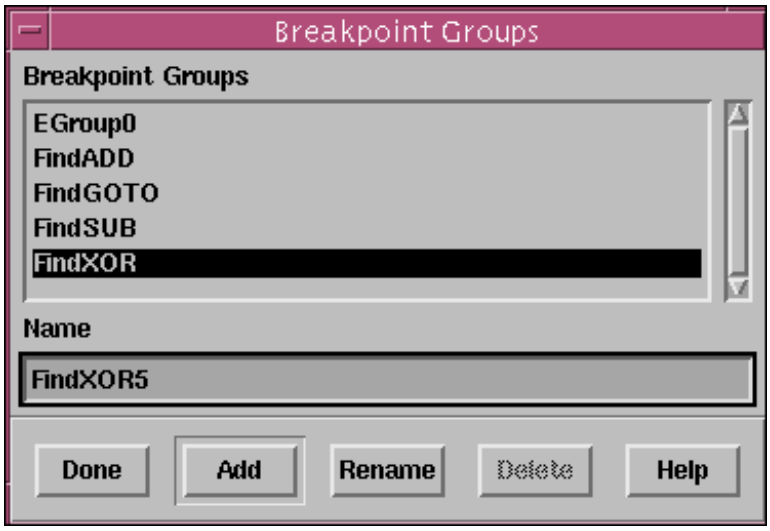
The following procedures describe how to create and use breakpoint groups. For specific information on creating expressions and triggers, see Chapter 15, Expressions.

- Creating Breakpoint Groups
- Adding Breakpoints to Breakpoint Groups
- Searching with Breakpoint Groups

Creating Breakpoint Groups

The Breakpoint Groups Dialog is used to add new breakpoint groups to the pane and Breakpoint Groups menus in the Waveform, Register, Source, and Logic Browser windows. Breakpoint groups are empty until expressions are assigned to them in the Expressions Dialog. (See Figure 4-7, Breakpoint Groups Dialog)

Figure 4-7 Breakpoint Groups Dialog



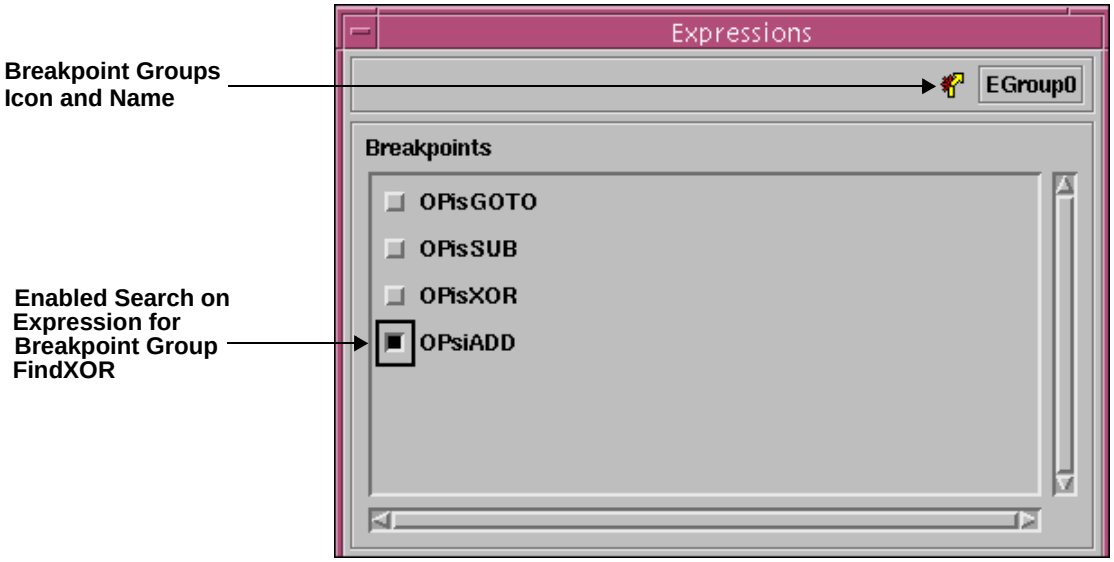
To create breakpoint groups:

1. Type the name of the breakpoint group in the Name field and click left on Add. The breakpoint group is added to the Breakpoint Groups Pane.
2. Click left on Edit in the menu bar and choose Breakpoints to open the Breakpoint Groups Dialog.
3. When you are finished, click left on Done to close the Breakpoint Groups Dialog.

Adding Breakpoints to Breakpoint Groups

After breakpoint groups have been created, the Expressions Dialog is used to define the contents of the breakpoint groups. A breakpoint group consists of a subset of all user-defined expressions. The Breakpoints Pane of the Expressions Dialog lists the full set of available expressions.

Figure 4-8 Expressions Dialog



To add expression breakpoints to a breakpoint group:

1. Click left on Edit in the menu bar and choose Expressions to open the Expressions Dialog.
2. In the Expressions Dialog, click left on the Breakpoint Groups icon to open the Breakpoint Groups Menu. The menu displays the breakpoint groups created from the Breakpoint Groups Dialog.
3. Choose a breakpoint group from the Breakpoint Groups Menu. The breakpoint group name is displayed to the right of the icon.
4. In the Breakpoints Pane, enable each expression you want to include in the breakpoint group. To enable or disable the expression, click left on the toggle box for the expression.
5. After you have enabled the expressions you want in this breakpoint group, either go to step 2 to work on a different group or click left on Done to close the Expression Dialog.

Searching with Breakpoint Groups

After creating a breakpoint group and defining expressions, you can use the breakpoint group to perform searches on enabled expressions. Each VirSim window includes a Breakpoint Group icon to open a menu of available breakpoint groups. All windows linked to the same Link group use the same breakpoint group.

1. In any window, click left on the Breakpoint Groups icon to open the Breakpoint Groups Menu and choose the desired breakpoint group. If the breakpoint group contains enabled expressions, the Search arrows in the toolbar are sensitized.
2. Click left on either the forward or backward search arrows to perform the breakpoint search. The C1 cursor stops at each breakpoint. In the previous example, select FindXOR and click left on forward search to locate the next XOR execution.

Collecting and Displaying Unique Events

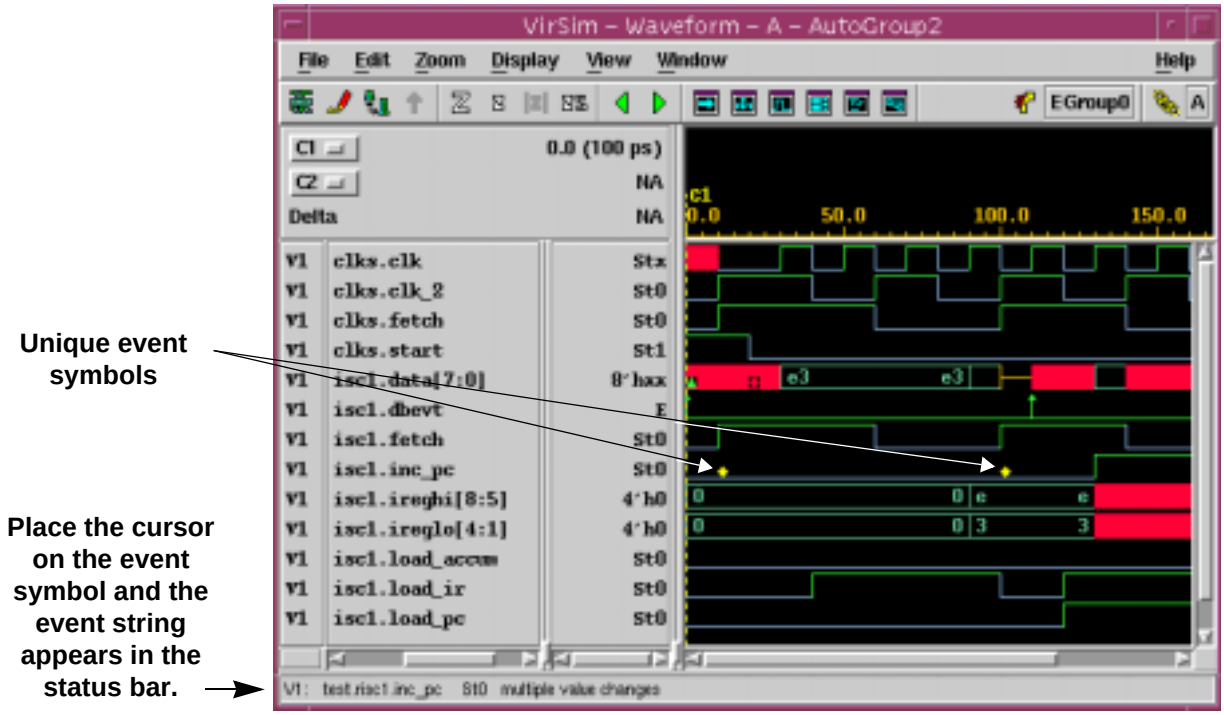
The Unique Events feature allows you to display symbols for unique events in the Waveform Window, Register Window, and Logic Browser. This is a Verilog only feature. By default, display of unique events is on. To turn off display of unique events, click left on the Display Menu in the menu bar and click Unique Events. Two types of unique events are currently available: glitch and user defined.

- Glitch events provide an indication that a signal has multiple value changes within one simulation time. They are always marked by a yellow diamond. When you place the cursor on the glitch event marker, the signal name and the message "multiple value changes" appear in the status bar.

- User defined events are defined using the $vcdplusevent command (refer to System Tasks and Functions on page 17-3). You can specify the type of event as a text string and the shape and color of the event marker for up to 244 different combinations. When you place the cursor on the event marker, the signal name and the event string appear in the status bar.

To enable glitch unique events in interactive mode, toggle recording of unique events in the Interactive Window by selecting Record Glitch Events from the Sim Menu. Actual recording starts at the beginning of the next time sample. If you are at or past the time for which you want to record glitch events, turn on Record Glitch Events then from the Sim Menu select Re-Exec to rerun the simulation. In post-processing mode you must enable reporting of unique events in the VCD+ file using $vcdplusglitchon for glitch events and $vcdplusevent for user defined events. For more information, refer to System Tasks and Functions on page 17-3.

Figure 4-9 Displaying Unique Events



Changing the Display Format

The following information is covered in this section:

- Selecting Drawing Mode
- Modifying Waveform Height
- Adding and Deleting Blank Lines
- Displaying Strength Based Colors
- Using the Waveform Style Editor

Selecting Drawing Mode

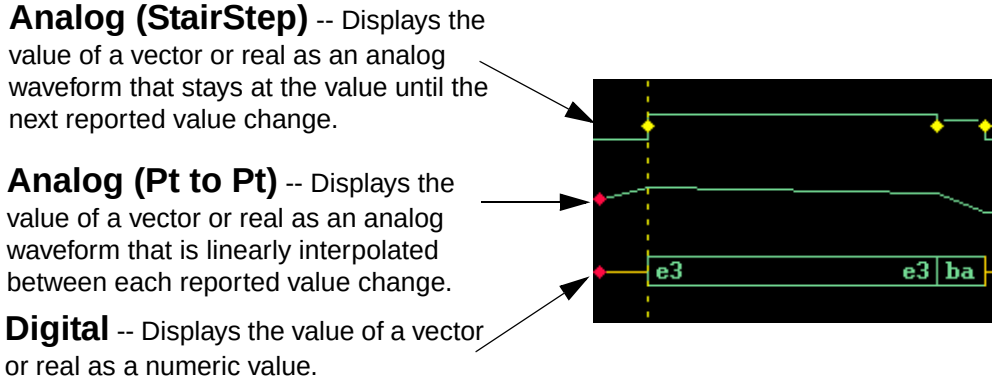
(EPIC and Verilog only) The drawing mode lets you display a vector in either digital or analog mode.

To select the Drawing mode:

1. In the Waveform Window, click right on the waveform to open the context sensitive menu.
2. Choose Drawing Mode and select the drawing mode type from the context sensitive menu.

There are three drawing modes: Analog (StairStep), Analog (Pt to Pt), and Digital. See Figure 4-10, Drawing Modes (EPIC and Verilog Only). Analog displays are typically used for EPIC signals, but can also be used to display vectors and reals as analog signals. In analog modes, the bottom of the signal wave represents the minimum value, and the top of the signal wave represents the largest possible value.

Figure 4-10 Drawing Modes (EPIC and Verilog Only)


Analog (StairStep) -- Displays the value of a vector or real as an analog waveform that stays at the value until the next reported value change.
Analog (Pt to Pt) -- Displays the value of a vector or real as an analog waveform that is linearly interpolated between each reported value change.
Digital -- Displays the value of a vector or real as a numeric value.
e3
e3
ba

Modifying Waveform Height

To modify the waveform height:

1. In the Waveform Window place your cursor on a waveform or select a group of waveforms and place your cursor on any one of the waveforms. Click right to open the context sensitive menu.
2. Select Waveform Height and then choose a multiple (from 1 to 20) of the standard height. The current size is grayed out. Your selection is applied to all selected waveforms.

Adding and Deleting Blank Lines

Blank lines are used to vertically space waveforms to improve the readability of waveforms in the Waveform Pane.

To insert a blank line above a waveform:

1. Click right on the waveform to open the context sensitive menu.
2. Select Blank. A blank line appears above the waveform.

To delete a Blank Line:

1. Click left on the blank space in the Signal Name Pane to select the blank line.
2. Select Cut or Delete from the Edit menu.

Displaying Strength Based Colors

The Waveform window can display different colors for different strengths. You display strength colors by incrementally loading the strengths.cfg file that is located in the $VIRSIMHOME/virsims_support/appfiles directory.

UNIX users can also display strength colors by setting the VirSim*enableStrengthBasedEncoding X resource to True. If your environment is set up so that XUSERFILESEARCHPATH is set to `./%N:~/%N:$XUSERFILESEARCHPATH`, you can do this by copying the $VIRSIMHOME/appfiles/VirSim file to either your current or home directory and editing the following line:

```
Virsim*enableStrengthBasedEncoding: True
```

Upon loading `strengths.cfg` or changing the resource, the default colors for strengths are automatically displayed as follows:

supply0, supply1	tan
strong0, strong1	green
pull0, pull1	purple
large0, large1	magenta
weak0, weak1	cyan
medium0, medium1	blue
small0, small1	pink
highz	gold

The difference between 0 and 1 strengths is the position of the line, for example, supply0 is a tan line at the bottom, supply1 is a tan line at the top.

You can change the default strength colors using the waveform style editor. You can, for example, use this editor to specify different colors for supply0 and supply1. Then, for future use, you can save a configuration file to incrementally load your preferred color scheme.

Using the Waveform Style Editor

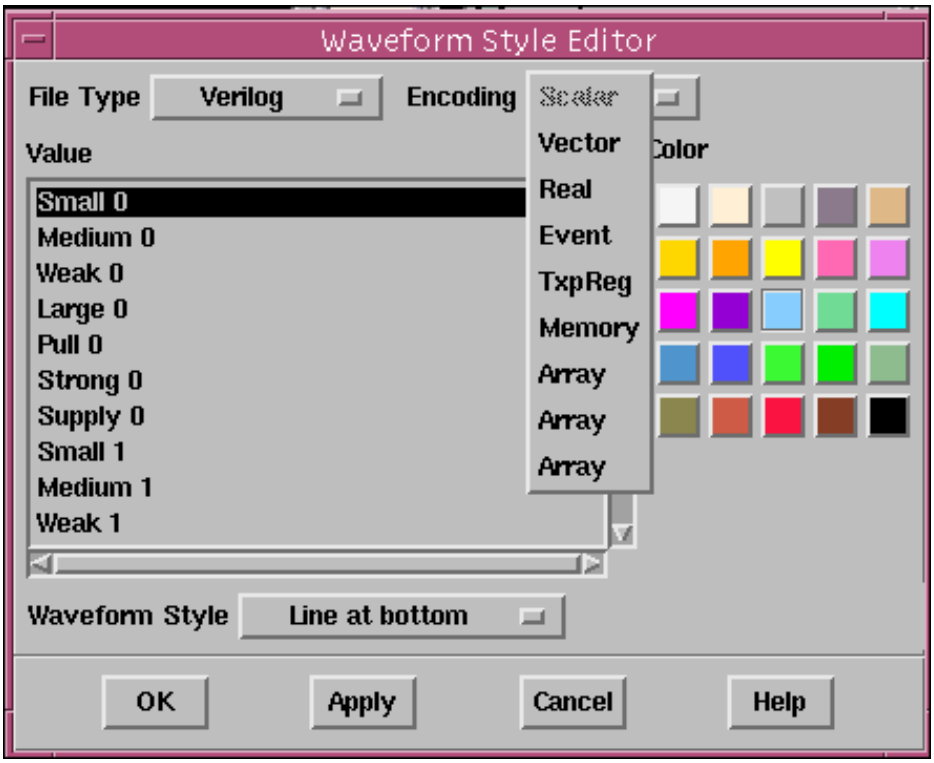
The Waveform Style Editor is used to modify the waveform color and style for selected encoding types. First, the file type determines the available encoding types. Then, the encoding type determines the available signal types (for example, values 0, 1, X, and Z for Verilog Scalar). Finally, for each signal type, its associated color and style can be set. The style settings can be saved in the VirSim configuration file and restored upon subsequent loading of that configuration file.

Default Waveform Styles for Verilog, VHDL, and EPIC are described in Appendix A, Waveform Defaults.

Note:

The default waveform colors and style settings are defined in the X Resources file. For information on changing the default settings, see Waveform Highlighting in Chapter 16, VirSim Setup.

Figure 4-11 Waveform Style Editor



The Waveform Style Editor has the following style menus and action buttons.

File Type

The File Type Menu is used to select the file type: Verilog, VHDL, or EPIC. Each file type has a unique set of options for encoding signal types, signal values, and waveform styles from which you can select.

Encoding

The Encoding Menu is used to select signal types for the Verilog, VHDL, or EPIC file you select. Verilog signal types include Scalar, Vector, Real, and Event. EPIC signal types include Scalar, Vector, and Analog. VHDL signal types include all standard types and all user defined types in all VHDL files that are currently open.

Value

The Value Menu is used to select the signal value. The Value Menu contains the available signal values for the selected signal type. When you select a value, the defined color and waveform style are displayed in the Color Palette and Waveform Style Menu.

Color

The Color Palette is used to set the signal color of the currently selected signal value and signal type. Colors identify the signal value and signal type. When you select a signal value, the palette displays a black border around the color assigned to the signal.

Waveform Style

The Waveform Style Menu is used to set the display style for the waveform. The file type, encoding type, and signal value determine the available set of waveform styles. When you open the menu, the current waveform style is desensitized.

OK

OK changes waveform styles to the styles set and closes the Waveform Style Editor.

Apply

Apply changes waveform styles to the styles set and leaves the Waveform Style Editor open.

Cancel

Cancel closes the Waveform Style Editor without saving any changes.

Help

Help opens online help for the Waveform Style Editor.

Additional Features

The following information is covered in this section:

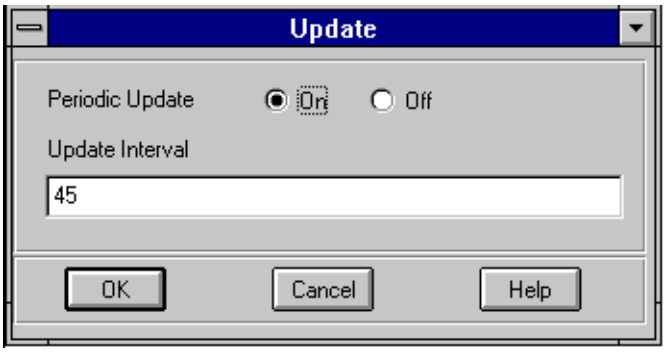
- Monitoring a Running Simulation Using Update Mode
- Printing Waveforms

Monitoring a Running Simulation Using Update Mode

The Update Dialog (see Figure 4-12, Update Dialog) is used to control periodic reading and display of new data. This allows VirSim to view results when either running an interactive session or reading a history file while the simulator is writing to that file.

To open the Update Dialog, click left on File in the menu bar and choose the Update command. This command is activated only during an interactive session or when reading a VCD+ file that is being written.

Figure 4-12 Update Dialog



The Update Dialog has Periodic Update buttons, an Update Interval field, and three action buttons.

Periodic Update

Toggles periodic update on or off. Click left on the appropriate toggle button to enable or disable periodic update.

Update Interval

Specifies the update interval (in seconds) to check the simulation history file for new data.

OK

Accepts the entry made in the Update Interval field and closes the Update Dialog.

Cancel

Closes the Update Dialog without saving changes.

Help

Opens online help for the Update Dialog.

Printing Waveforms

To access the Print Dialog from the Waveform Window, click left on the File Menu and choose Print. If multiple Waveform Windows are displayed, the Print Dialog simply associates with the Waveform Window from which it is selected.

When using the default Print Dialog settings, printed output is very similar to what is viewed in the associated Waveform Window, with the following differences.

- Time-axis display is more elaborate. Default time tick intervals are always {1, 2, or 5} x 10n and are chosen by VirSim to maximize the readability and reference value of the output. Time values may also be vertically staggered in certain cases to allow for more frequent labels without loss of readability. Also, vertical grid lines are lightly drawn through the data area to provide more visual cues.
- All signals in a Group are included in a print job, even those that are not directly visible in the Waveform Window.
- The Value Pane of the Waveform Window is not printed.
- The Print Dialog provides time display options, which provide user-control of display and scaling of time-axis values.

This section includes the following information:

- Printing a Time Range
- Distributing a Printout Across Several Pages
- Controlling Paper Size and Orientation
- Specifying a Printer Name

- Printing to Postscript
- Adjusting the Signal Name Printout

Printing a Time Range

Use the Begin Time and End Time fields to specify the time range to print. By default, Begin Time displays time at the left edge of the Waveform Pane and End Time displays time at the right edge. The controls automatically constrain entries to times within the simulation.

Distributing a Printout Across Several Pages

Use the Time Slices and Signal Slices controls to distribute printouts across pages.

To print the time axis across several pages:

Use Time Slices to divide the time (x) axis into up to 50 slices (pieces) and print a slice on each page. For example, to print from time 1000 to time 2500 across four pages, set Begin Time to 1000, End Time to 2500, and Time Slices to 4. Each page in the printout has (2500 - 1000) / 4 = 375 time units.

To print the signal axis across several pages:

Use Signal Slices to divide the signal (y) axis into up to 20 slices (pieces) and print a slice on each page. If the last time slice is only partial, the remainder appears on the last page. For example, if you are printing a group with 37 signals, and Time Slices is set to 1, a request for 4 Signal Slices results in 10 signals on the first three pages and 7 on the last. If Signal Slices is set to one, all 37 signals appear on one printed page. If there are 100 signals in the current group, and Time Slices is one and Signal Slices is 4, 25 signals will be printed per page.

You can print with both multiple time slices and multiple signal slices to break a job up into as many as 1000 (50 time slices * 20 signal slices) sheets. Each page of the output is numbered (for example, 5 of 12).

Controlling Paper Size and Orientation

To select a standard paper size:

Select the Standard Paper Size button then select one of the following sheet size options from the associated pull-down menu: American standard A, B, C, D, or E size sheets, or Metric Standard A3, A4, or A5 sheets.

To specify a custom sheet size:

Select the Custom Sheet Size button then enter values for Width, Height, and Units. Width and Height fields. You can select units in inches (in), centimeters (cm), and millimeters (mm) from the Units pull-down menu.

To rotate the output:

Many PostScript output devices assume a portrait orientation for their output space. By default, VirSim prints in landscape mode. Thus, it sets the Rotate Output toggle on. If you prefer portrait orientation, set the toggle off.

Specifying a Printer Name

For Windows users, use the Print Command field to specify a printer port such as LPT1 for a local printer or a path and printer name such as \\helga\engprinter for a network printer.

Note:

For Unix, the only command accepted in this field is lpr.

Printing to Postscript

To print to a PostScript file:

When you select the Print Only to File option, PostScript output is sent only to the specified file. This can be extremely useful in conjunction with a PostScript preview tool. For example, you can interactively adjust and view your print job prior to generating hardcopy by making print option adjustments, pressing the Apply button to generate a new PostScript file, then reading the file in your PostScript previewer. You can view the exact print job before submitting it to the printer

Note:

If you do not select Print Only to File, Apply sends the print job to the printer instead of a PostScript file.

To create an Encapsulated PostScript file:

The Encapsulated PostScript (EPSF) toggle button is associated with the Print To File toggle button, and has the sole effect of centering the PostScript image on the page with margins equal to 25% of the page width and height. The generated PostScript is fully scalable (i. e. all graphics are drawn relative to the image origin), and may be relocated anywhere on the page using whatever tools are provided by the importing application. The PostScript image emitted using Print To File is fully scalable even without this option, but the margins are set such that the image occupies most of the page. The Encapsulated PostScript option outputs an image that is more easily manipulated in the importing application.

Adjusting the Signal Name Printout

To print the local signal name or hierarchical signal name:

Select Local Name or Full Name from the Signal Name pane.

To adjust the width of the signal name in the printout:

Select Shrink to Fit to automatically size the signal name column to fit the longest signal name. To specify a width for the signal name column, select Signal Width and enter a value in the Width field. You can select units in inches (in), centimeters (cm), and millimeters (mm) from the Units pull-down menu.

Toolbar and Menu Reference

Toolbar

You can toggle display of toolbar icons from the View menu.

Group Icon	The Group icon opens a menu containing all defined signal groups. The five most recently used groups appear at the top of the menu, with the most recent group first. The remaining groups are listed in alphabetical order in the lower portion of the menu. Selecting a group loads the signals for that group and displays the group name in the title bar of the Waveform Window. For more information on working with groups, see Using Multiple Signal Groups to View Relevant Data.
Marker Icon	The Marker icon opens a menu containing all defined markers. In post simulation mode, selecting a marker from the menu moves the C1 cursor to the set marker. If the zoom was saved, selecting he marker restores the zoom factor. In Interactive mode (SIM Link), the marker advances the simulator to the time selected in the linked window. For more information on markers, see Chapter 12, Markers.
Load Icon	The Load icon can be used during Update mode to manually read any new information that has been written to the VCD+ file.
Zoom In Icon	The Zoom In icon zooms in the Waveform pane to display approximately 50% of the previous time range in the Waveform Pane. The zoomed display centers around the C1 cursor.
Zoom Out Icon	The Zoom Out icon zooms out the Waveform Pane to display approximately 2005 of the previous time range in the Waveform Pane. The zoomed display centers around the C1 cursor.

Zoom Cursors Icon	The Zoom Cursors icon zooms the Waveform Pane to display a time range between cursor C1 and C2. To avoid rounding errors, the Waveform Pane may display a larger time range than that marked by cursors C1 and C2. You can also zoom the cursors by right clicking in the Timescale Pane or selecting Zoom Cursors from the context sensitive menu.
Zoom Percent Icon	The Zoom Percent icon displays a pop-up menu of zoom percentages: 100%, 75%, 50%, and 25%. When you select one of these options, the Waveform Pane zooms to that percentage of the simulation time. Percentages are based on the minimum and maximum simulation time range of signals for all data files loaded.
Vertical Compress	The Vertical Compress icon toggles between normal and condensed waveforms. Condensed waveforms provide an overview mode which displays as many waveforms as possible with enough detail to determine what areas need further investigation.
Search Icons	The Search icons initiates a search backward or forward in time from the C1 cursor to the next breakpoint. The search includes active expression breakpoints and line breakpoints in any Source Window in the Waveform Window. If the search finds a breakpoint or event for an active expression, C1 is placed at the event in the Waveform Pane. The name of the expression satisfied is displayed above C1. If multiple expressions are satisfied at the same simulation time, VirSim displays an asterisk and the name of the first expression. Additional searches in the same direction cycle through the remaining expression names. No names are displayed for line breakpoints. If the search does not find any events for any active expression, the system bell sounds.

 New Window Icons	These icons allow you to open new VirSim windows from the Waveform Window toolbar.
 Breakpoint Group Icon	The Breakpoint Group icon opens the Breakpoint Groups menu. Breakpoint Groups consist of groups of expressions that are used to search for breakpoints. You can select available breakpoint groups from the menu. Breakpoint groups are created in the Breakpoint Groups Dialog. The group expressions are defined in the Expressions Dialog. For information on creating Breakpoint Groups, see Monitoring a Running Simulation Using Update Mode.
 Link Icon	Windows may be linked so that changes in time in one window are reflected in all other windows. The linked windows display the current link as a capital letter or SIM next to the Link icon. To link or unlink a window, click the Link icon and select the desired link. Moving or placing Cursor C1 changes the time of the other linked windows. In the case of an interactive simulation, all windows linked to SIM are linked to the current time in the Waveform Window. In this case, Cursor C1 is used only for Zoom Cursors.

Context Sensitive Menus

Signal Group Pane Context Sensitive Menu (CSM)

The Waveform Pane CSM is used to display and format waveforms, expand and collapse times, and obtain event origins (see Figure 4-13, Waveform Pane CSM). To open the Waveform Pane CSM, click right on the waveform or a waveform edge.

Figure 4-13 Waveform Pane CSM

Zoom Cursors

Displays a waveform time range between cursors C1 and C2.

Event Origin

Brings up a sub menu of Event Origin options: Automatically Select Window, Point to Preferred Window, View In New Window. Refer to Chapter 12, Event Origin.

Drawing Mode

Brings up a menu of drawing mode options: Digital, Analog (StairStep), and Analog (Pt to Pt). Refer to Selecting Drawing Mode on page 4-30.

Waveform Height

Allows you to choose a multiple (from 1 to 20) of the standard height. The current size is grayed out.

Add Blank

Places a blank line above the waveform on which you clicked to open the CSM. To delete the blank line, click left on the blank space in the Signal Name Pane to select the blank line then select Cut or Delete from the Edit menu.

Add Drivers

Allows you to display the drivers for a signal either directly below the signal or at the end of the current group.

Expand Time

Expands a time to view delta cycle information. Refer to Expanding Time to Display Exact Order of Events Using Delta Cycles on page 4-19.

Collapse Time

Collapses delta cycle information for the time on which you clicked to open the CSM. Refer to Expanding Time to Display Exact Order of Events Using Delta Cycles on page 4-19.

Collapse All Time

Collapses all expanded delta cycle information. Refer to Expanding Time to Display Exact Order of Events Using Delta Cycles on page 4-19.

Next Edge

Locates the next edge that occurs in any of the selected signals. Refer to Finding the Next or Previous Edge on Selected Signals on page 4-17.

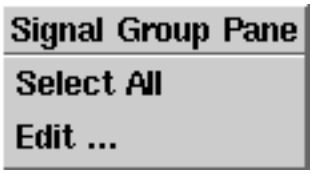
Previous Edge

Locates the previous edge that occurred in any of the selected signals. Refer to Finding the Next or Previous Edge on Selected Signals on page 4-17.

Signal Pane Context Sensitive Menu (CSM)

The Signal Pane CSM is used to select all signals in the pane. (see Figure 4-14, Signal Pane CSM). To open the Signal Pane CSM, click right in the Signal Pane.

Figure 4-14 Signal Pane CSM



Select All

Selects all signals in the Signal Pane.

Edit ...

Opens a dialog that allows you to edit the name or bit range.

Menu Bar

The following menus include commands specific to the Waveform Window:

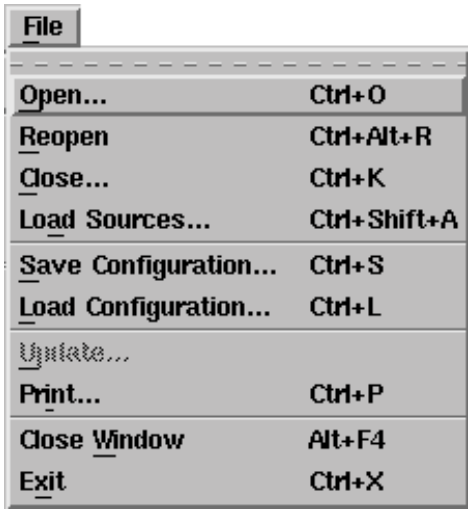
- File Menu
- Edit Menu
- Zoom Menu
- Display Menu
- View Menu

Note:

Any menu with a dotted line at the top can be torn off and placed on the display for easy access. Click middle above the dotted line, and then drag the menu to a convenient position.

File Menu

Open - Opens the file browser to load an EPIC, VCD, or VCD+ history file.
Reopen - Reopens all history files that have changed since they last were loaded.
Close - Closes an open history file.
Load Sources - Compiles Verilog source.
Save Configuration - Saves the current configuration.
Load Configuration - Loads a configuration file.
Update - Opens the Update Dialog to enable or disable periodic reading of history files from an active simulation or, in interactive mode, a running simulation.
Print - Opens the Print Dialog for printing signal data.
Close Window - Closes the Waveform window.
Exit - Exits VirSim.



Edit Menu

Cut - Cut the selection and move it to the clipboard.

Copy - Copy the selection to the clipboard.

Paste - Paste the clipboard to a designated area.

Delete - Delete the selection.

Select All - Selects all signals in the window.

Groups - Open the Groups Dialog. A group is a set of signals that are global to all Waveform windows. You create groups in the Groups Dialog.

Markers - Open the Markers Dialog. The dialog is used to define markers for specific simulation times.

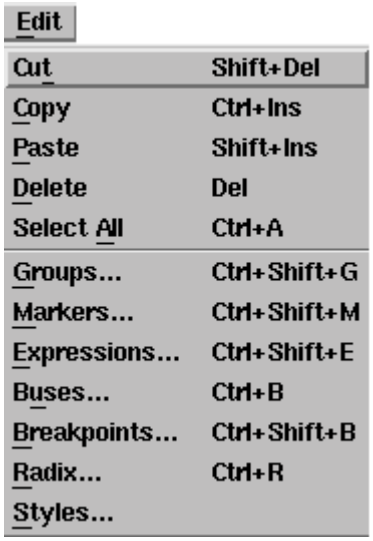
Expressions - Open the Expressions Dialog. The dialog is used to define and enable expressions for breakpoint groups.

Buses - Open the Bus Builder Dialog.

Breakpoints - Open the Breakpoint Groups Dialog. The dialog is used to create breakpoint groups for grouping breakpoint expressions.

Radix - Open the Radix Dialog. The dialog is used to create user-defined radices.

Styles - Open the Waveform Style Editor Dialog. In the dialog, you can set waveform styles for the file type, encoding, color, and waveform position and shape.



Zoom Menu

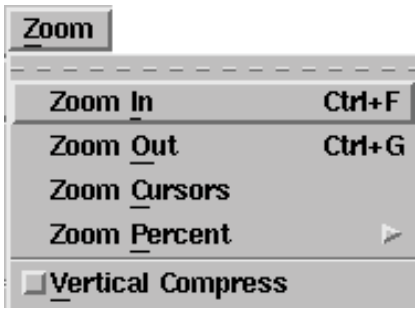
Zoom In - increases waveform detail. It displays approximately 50% of the previous time range.

Zoom Out - increase waveform viewing area. It displays 200% of the previous time range.

Zoom Cursors - displays a waveform time range between cursors C1 and C2.

Zoom Percent - Displays a popup menu to choose Zoom Percent of the total time range: 100%, 75%, 50% and 25%

Vertical Compress - toggles compression of display vertically.



Display Menu

Time Scale - Opens the Time Scale Dialog which is used to set the display unit and display precision for VirSim display time.

Unique Events - Toggles Unique Events *on* or *off*. When *on*, unique events display in the Waveform Window. When *off*, unique events do not display.

View Menu

View Menu settings are saved when you close VirSim.

Waveform Tools - Toggles display of the Waveform tools.

Zoom Tools - Toggles display of the Zoom tools.

Search Tools - Toggles display of the Search tools.

Window Icons - Toggles display of the Window icons.

Break Group Icon - Toggles display of the Breakpoint Group icon.

Link Icon - Toggles display of the Link icon.

5

Source Window

Using the Source Window you can:

- Navigate and view the design and transparently access all its elements in complex file hierarchies.
- Easily edit the correct file using your preferred editor.
- Debug design behavior with current signal values annotated next to the signals.
- Use Event Origin to locate the origin of any signal change.
- Follow and control source execution in selected sections of the design using breakpoints and stepping.
- Debug design behavior by analyzing execution counts for the code.

Introducing the Source Window

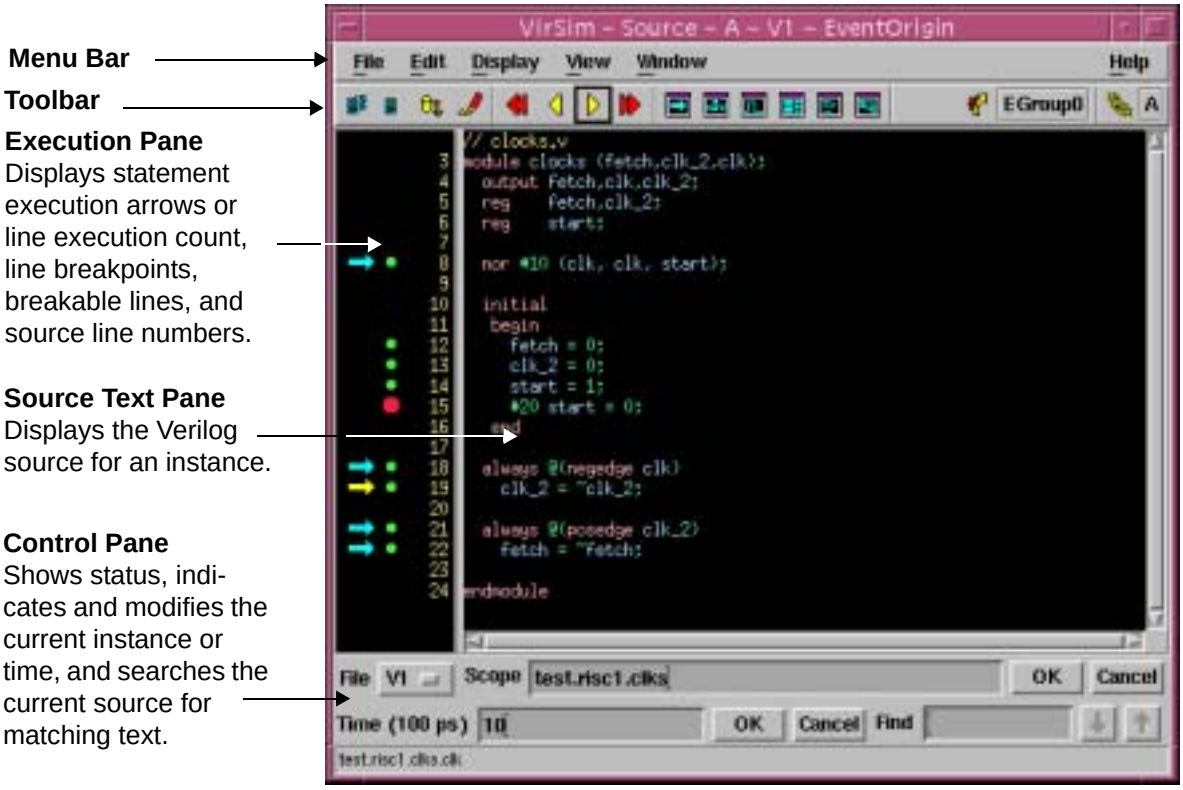
The Source Window has the following three panes:

- Execution Pane
- Source Text Pane
- Control Pane

In addition, there is a toolbar for commonly used commands.

Figure 5-1, Sample Source Window, shows an example of the Source Window.

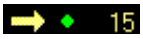
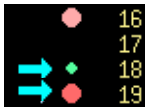
Figure 5-1 Sample Source Window



Execution Pane

The Execution pane normally displays arrows to indicate statement execution and dots to indicate line breakpoints (see Table 5-1, Statement Indicators in Execution Mode).

Table 5-1 Statement Indicators in Execution Mode

Indicator	Description
	In Post Simulation mode, small green dots represent lines that have been executed at least once in the simulation and where a breakpoint might be set. In Interactive mode, the small dots indicate lines that may potentially execute.
	For a given simulation time, multiple arrows may be displayed to indicate concurrent statement execution.
	A single yellow arrow indicates the current statement execution. In Post Simulation mode, only one current statement per linked Source Window can execute at a time. You can single step the arrow forward and backward.
	A red dot is displayed on lines where breakpoints have been set. Pink dots indicate breakpoints se in other instance groups. For VHDL, pink dots also indicate breakpoints set from the command line of the Interactive Window. To set or clear a breakpoint, left click on the dot.
 	(Verilog only) An off-white outline arrow marks the results of an Event Origin operation or indicates source text that resulted from dragging and dropping any of the following from the Logic Browser into the Source Window: • Assignment object • Module instance object • Primitive graphical object A filled yellow arrow indicates the Event Origin or source text is also the current execution file.

Alternatively, the Display menu commands Show Execution and Show Coverage toggle the Execution pane between execution and coverage display modes. Selecting Show Coverage displays the line execution count in place of the statement execution arrows. Selecting Show Execution displays the arrows again.

In order to obtain line info and/or line trace data, the simulator may have to be built with or supplied appropriate options or commands.

In Interactive mode when the All group is enabled, line execution data is not saved. There will be no hollow blue arrows indicating concurrent statement execution.

Source Text Pane

The Source Text pane displays the Verilog source code for the instance. To select an instance, either

- enter the full name in the Scope field and left click on the **OK** button, or
- drag-and-drop a scope to the field from another VirSim window.

When you drag a scope into the source text pane, an instance is added to the currently enabled instance group. If no instance groups have been created, VirSim creates and automatically names a group (AutoInstanceGroupX, where X is numeric).

You can drag-and-drop signal names from the Source Text pane to other windows or dialogs. To select text, double click on the text or left click and drag the cursor across the text.

The source text is color coded to designate user identifiers, keywords and comments.

Color	Use
Blue	User-defined identifiers
Red	Keywords
Yellow	Comments
Green	All other text
Gray	Defined or protected source

Note:

Colors shown above are the ones defined in the standard configuration.

Control Pane

The control pane includes the following controls (see Table 5-2, Control Pane).

Table 5-2 Control Pane

Control	Description
File	Specifies the file designator for the source history file or compiled sources (Z1 file). The file designator is selected from the pull-down menu next to the File field. NOTE: For files created with non-VCS and Scirocco simulators, only VCD+ history files (or interactive session) that have Use Sources enabled appear in the File menu.
Scope	Specifies the full name of the scope loaded in the Source Text pane. When you enter a scope, the Source Window displays the instance of that

Table 5-2 Control Pane

Control	Description
Time	Specifies the time of the window and al windows linked to this window.
Find	Searches the current source for matching text. The Find field includes a text entry field and forward and backward search buttons. If the signal is dragged into the Source Window, the Find box is seeded with the signal name and automatically highlights the first occurrence. To search text, enter the character string to search for in the text entry field and left click on the search button. Note: - The Find field does not support wildcard characters. - Search text does not search text spanning line breaks. - If a forward search reaches the end of the instance before finding text, the search continues at the beginning of the instance. - If no match is found, the status bar displays an error and the console beeps.
Status bar	Displays a description of the toolbar buttons and informative messages.

Before Using the Source Window

Verilog

Verilog source information can come from either the Verilog source or from the simulator PLI for certain simulators. The VCD+ file produced by VCS or Scirocco when compiled with the -line option contains information that allows the VirSim Source Window to work without first compiling the source. When the VCD+ file is from other simulators, the Verilog source must be compiled by VirSim. The Load Sources dialog will automatically pop up if VirSim needs to compile sources and source file information was not provided on the command line. The source files do not necessarily need to reside in your local directory; however, if they do not, you must provide the full path. You may provide the source arguments on the VirSim command line, and the compilation then will be performed when the information first is required.

Line trace information can come from either a VCD+ file or via a PLI from an interactive run. For line trace data to be stored in the VCD+ file, the simulator must have been explicitly instructed to store line trace information in the Verilog code using $vcdplustraceon. When the information is coming from a VCD+ file created with a simulator other than VCS, the VCD+ file also must have been opened with the Use Sources switch enabled (which is the default).

VHDL

You need to analyze the source code in debug mode before running simulation.

Note:

You can not display the cycle portion of your design in the source window.

Navigate and View a Design

Drag-and-drop a scope from any other window to the Source Window, and the Source Window displays source for that instance module. An instance can be any module, task, function, or named block in the Verilog source code. Instances may be placed in groups so that a group of instances may be debugged. A special All Group instance group allows source debugging of the entire design in interactive mode or those portions for which line trace data was saved in post-processing mode.

Use the context sensitive menu to descend new instances or to go to the parent of the currently displayed instance. Each instance visited is added to the current instance group for use when following or controlling source execution.

Access any signal or signals from the Source Window for use in any other window by dragging and dropping individual signals or a selected range of code to another window. Signal hierarchy is added to the names, signals are sorted and duplicates are removed.

Edit the Correct Design File

The Source Window Edit menu includes two commands for opening source code in a text editor: Edit Source and Edit Parent.

To	Do This
Open the Source File for the current instance	In the Menu Bar, left click on Edit and choose Edit Source. For example, if the current instance displayed is test.risc1.instdec, then the text editor displays source code for test.risc1.instdec.
Edit a specific line of source code or an included file.	Left click on the line in the Source Text pane and then from the menu bar choose Edit and Edit Source. This also allows editing included files.
Open the source file for any parent instance of the currently displayed instance.	In the Menu bar, left click on Edit and choose Edit Parent. For example, if the current instance is test.risc1.instdec, then the text editor will display source code for test.risc1.

Note:

If you change the source file, you must resimulate.

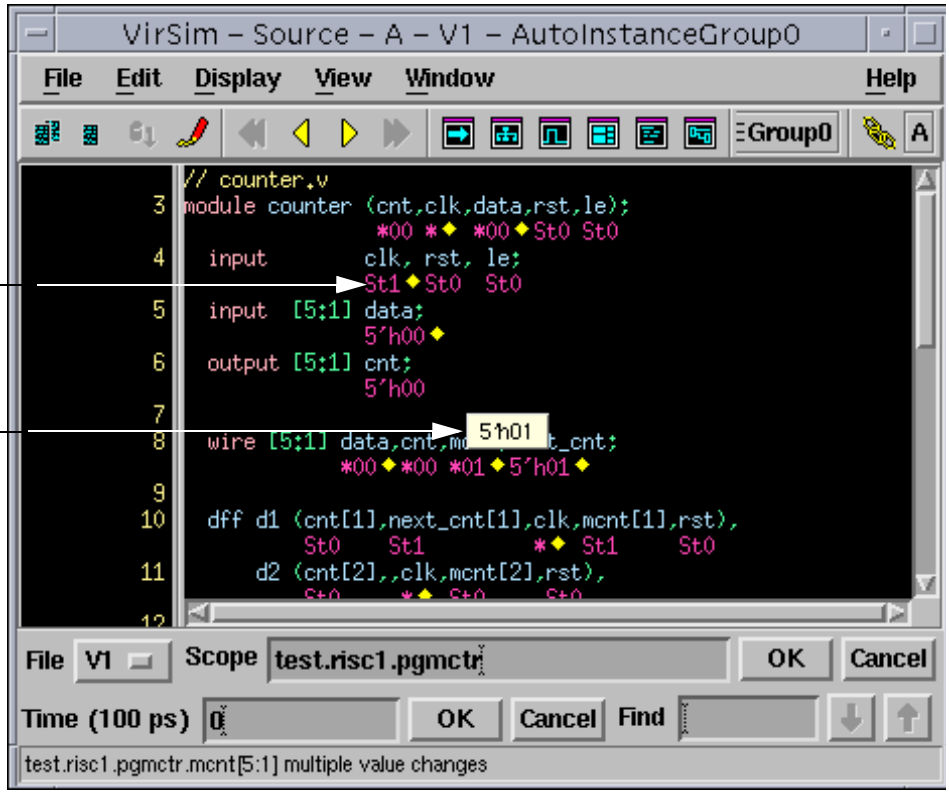
Debug Behavior Using Value Annotation

View Signal Values for All Signals

To view current values for all signals, select Show Values from the Display menu. Current signal values appear below the variable name.

If there is not enough space for a signal value to appear, an asterisk appears next to the signal name. Move your cursor over the signal name to view a popup of the current signal value.

Figure 5-2 Source Window with Show Values Turned On



Current signal values appear below the variable name

An asterisk appears when there is not enough space to display the value. Move the mouse over the variable to view a value popup

View Signal Values for One Signal

To view the current value for one signal, click the Load Value Changes icon if necessary, then move your cursor over the signal name to view a popup of the current signal value.

Follow and Control Source Execution Using Stepping and Breakpoints

Follow Source Execution

The Source Window displays arrows to indicate all lines that execute this sample time and what line will execute next. Source execution is followed for all instances in the current instance group.

Control Source Execution

Use the Next/Previous Line icons to follow execution. You can also use the Run icon or any other means to change the current sample time to see all lines that execute at a sample time.

Using Breakpoints

Set Breakpoints

In the Execution pane, green dots next to a statement line indicate lines that can have breakpoints. Each instance group has its own set of breakpoints. Setting a breakpoint on a line of an instance only affects the current instance group. When All Group is being used, pink dots indicate that a breakpoint is set in one or more of the other instance groups. To set a breakpoint, left click on the green or pink dot. A large red dot replaces the dot, indicating a breakpoint has been set at that line.

Clear Breakpoints

The Edit menu contains two commands for clearing breakpoints: Clear Instance Breaks and Clear Group Breaks.

- The Clear Instance Breaks command removes all line breakpoints in the instance currently being shown.
- The Clear Group Breaks command removes all line breakpoints in all the instances in the instance group currently in use.

In the menu bar, left click on Edit and choose the appropriate breakpoint command.

To clear individual breakpoints, left click on the red breakpoint dot. The red dot is replaced by a green dot. If a pink dot is displayed, you must go to the instance group which set the breakpoint.

Run to Breakpoints

The Step Breakpoint icons set the Source Window time to the previous or next time a line containing a breakpoint executed. When All Group is selected, all lines for which a breakpoint has been set in any group are considered. Otherwise, only those lines in the current instance group with breakpoints set are considered.

Note:

VirSim might stop before a line containing a breakpoint if it encounters an expression breakpoint.

Using Instance Groups

Instance groups are used to filter trace execution information. Only statements executed in instances listed in the current instance group are displayed.

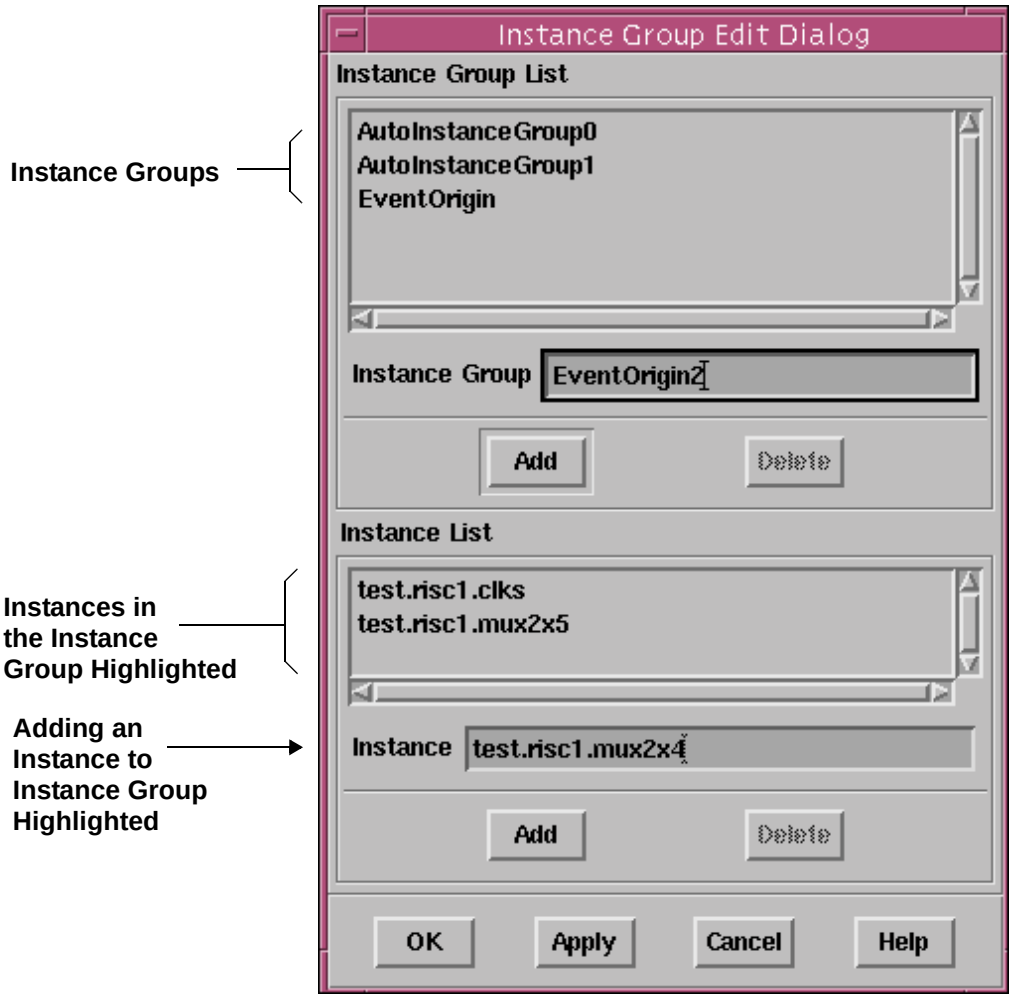
The Instance Group Edit Dialog is used to create and edit instance groups. You can add instances to the group by dragging them into the Instance List. You also can add instances to the current instance group by dragging them into the Source Window.

For example, If you are debugging a problem, you can create and use an instance group that contains only those instances in which the problem may occur. When you step lines, the Source Window only steps through instances in the order executed.

A special All Group instance group allows source debugging of the entire design in interactive mode or those portions for which line trace data was saved in post-processing mode.

Figure 5-3, Instance Group Edit Dialog, shows an example of the Instance Group Edit Dialog. To open the Instance Group Edit Dialog, left click on Edit in the Menu Bar and choose Instance Groups.

Figure 5-3 Instance Group Edit Dialog



The Instance Group Edit Dialog has the following options.

Instance Group List

Lists the names of the current instance groups.

To display the instance group:

- left click on an instance group in the Instance Group List. The instances in the group appear in the Instance list.

To add instances to the instance group:

- with the instance group highlighted, enter a new instance in the Instance field and left click on **Add**, or
- drag-and-drop a scope into the Instance List pane.

Instance Group

Used to enter a new instance group. Type the name of the instance group in the text field and left click on **Add**. The name appears in the Instance Group List.

Instance List

Lists the names of instances in the instance group. To display the instance list, left click on an instance group in the Instance Group List.

Instance

Used to enter an instance in an instance group. To enter an instance, either:

- type the full hierarchy name of the instance in the Instance text field and left click on **Add**, or
- drag-and-drop the instance into the instance name text field and left click on **Add**, or
- drag-and-drop the instance directly into the Instance list.

Add

Adds an instance to the Instance List or an instance group to the Instance Group List.

Delete

Deletes the selected instance from the Instance List or the selected group from the Instance Group List.

OK

Saves changes to the Instance Group Edit Dialog and closes the dialog.

Apply

Saves changes to the Instance Group Edit Dialog and leaves the dialog open.

Cancel

Closes the Instance Group Edit Dialog without saving any changes.

Help

Opens on-line help for the Instance Group Edit Dialog.

Define Markers

The Markers Dialog is used to define markers for simulation times. To open the Markers Dialog, left click on the Edit menu in the menu bar and choose Markers. For more information on markers, see Chapter 14, Source Window.

Capture Line Data

The Capture Line Data command is used to capture and display line execution data in the Interactive mode. To capture line execution data, left click on the Display menu and then left click on the toggle box to enable Capture Line Data.

By disabling capture of line data, you gain significantly improved performance when using line breakpoints in interactive Verilog-XL simulations. The default for Capture Line Data is on.

Note:

> When you set breakpoints from the Source Window, indications of which lines are breakable (green dots) may change. This is due to differences in VirSim's algorithm for determining breakable lines using the line trace method versus the simulator's ability to break at a given line (using the `$db_breakatline` command).

Note:

> When using the All Group in Interactive Mode, the Capture Line Data selection on the Display menu is insensitive as line data is not captured in this optional mode.

Debug Behavior by Analyzing Execution Counts

To display line execution counts, left click on Display in the menu bar and left click Show Coverage. This feature can provide valuable information when debugging. For example, it can tell you if the appropriate lines were executed or if inappropriate lines were executed.

Line stepping is disabled when Show Coverage is selected.

Toolbar and Menu Reference

This section includes the following information:

- Toolbar
- Source Text Pane Context Sensitive Menu
- Menu Bar

Toolbar

You can toggle display of toolbar icons from the View menu.

Instance Group	The Instance Group icon opens a menu of available instance groups. An instance group is a set of instances for which source tracing can be enabled. Left click on an instance group to select it. When the instance group is changed and the currently displayed instance is not in the new group, the first instance within the group is displayed. For information on creating instance groups, see Using Instance Groups. The All Group instance group is a special instance group that allows source level debug of the entire design. If running interactively and All Group is selected, line stepping will stop at any line in the model. When in Interactive Mode and using the All Group, line data is not saved in VirSim memory. Therefore, no line concurrent execution indication appears in the Source Window Execution Pane. Also, when switching to the show coverage view, the line would will not be implemented for lines executed while the All Group is active. In post-simulation, only those scopes which had line tracing enabled during simulation are active for line stepping.

Select Instance	The Select Instance Group icon opens an alphabetized list of all instances in the currently selected instance group. Left click on an instance to select it. The selected instance will then appear in the Source Text pane. Due to the size and ambiguity of the instance list, this icon is inactive when the All Group instance group is selected.
Load Value Changes	The Load Value Changes icon loads value changes for all signals in the current scope. Once loaded, you can display current signal values by positioning the mouse cursor over a signal name in the Source Window. After a short delay, VirSim displays the current value of the signal in a small pop-up window. The swLoadValueChanges X-resource allows value changes to be automatically loaded. If this is set to Auto, the Load Value Changes icon is insensitive.
Marker	The Marker icon opens a menu of defined markers. Left click on the Marker icon to set the Source Window to the time of that marker. In Interactive Mode (SIM link), the marker advances the simulator to the time selected in the linked window. For more information on markers, see Chapter 12, Markers.
Step Breakpoint	The Step Breakpoint icons set the Source Window time to the previous or next time a line contacting the breakpoint executed. These buttons are enabled when a line breakpoint is set. When All Group is selected, all lines for which a breakpoints has been set in any group are considered. Otherwise, only those lines in the current instance group with breakpoints set are considered.
Step Line	The Step Line icons set the current statement arrow to the previous or next statement execution. If there are no more statements to execute at the current time, the Source Window time is set to the previous or next time any statement was executed.
New Window Icons	These icons allow you to open new VirSim windows from the Source Window toolbar. You can toggle display of these buttons by selecting Window Icons from the View menu.

 Breakpoint Group	The Breakpoint Group icon opens the Breakpoint Groups menu. Breakpoint Groups consist of groups of expressions that are used to search for breakpoints. You can select available breakpoint groups from the menu. Selecting a breakpoint group only affects expression breakpoints. The line breakpoints remain the same. Different sets of line breakpoints may be activated with the selection of different instance groups. Breakpoint groups are created in the Breakpoint Groups dialog. The group expressions are defined in the Expressions Dialog. For information on creating breakpoint groups, see Locating Events Using Breakpoint Groups with Enabled Expressions.
 Link Icon	Windows may be linked so that changes in time in one window are reflected in all other windows. The linked windows display the current link next to the Link icon. The link indicator may either be SIM, which indicates the window is linked to the simulator, or a capital letter which indicates it is linked to all windows with the same link and not linked to the simulator. To link or unlink a window, click the Link icon and select the desired link. Unlike other analysis windows, only one Source Window per link is allowed. This window, however, can handle any combination of instances as defined in the instance group.

Source Text Pane Context Sensitive Menu

To open the CSM, point and click right in the Source Text pane.

Figure 5-4 Source Text Pane CSM

Descend Instance

Selects the new scope, adds it if necessary to the current instance group, and displays its source in the Source Text pane. This command is enabled when the word currently being pointed at if appended to the current instance results in a legal instance name for a module.

Go to Parent

Selects the parent of the currently displayed scope, adds it if necessary to the current instance group, and displays its source in the Source Window. The equivalent action is also taken by left clicking the Display menu and selecting Go to Parent.

Select All

Selects all signals in the Source Text Pane.

Event Origin

Brings up a sub menu of Event Origin options: Automatically Select Window, Point to Preferred Window, View In New Window. Refer to Chapter 12, Event Origin.

Menu Bar

Note:

Any menu with a dotted line at the top can be torn off and placed on the display for easy access. Click middle above the dotted line, and then drag the menu to a convenient position.

File Menu

Open - Opens the file browser to load an EPIC, VCD, or VCD+ history file.

Reopen - Reopens all history files that have changed since they last were loaded.

Close - Closes an open history file.

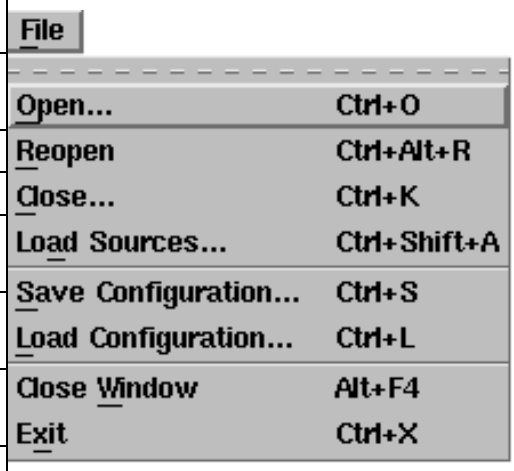
Load Sources - Compiles Verilog source.

Save Configuration - Saves the current configuration.

Load Configuration - Loads a configuration file.

Close Window - Closes the Source Window.

Exit - Exits VirSim.



Edit Menu

Copy - Copies the selection to the clipboard.

Select All - Selects all signals in the window.

Edit Source - Opens the source file for the current scope in a text editor window.

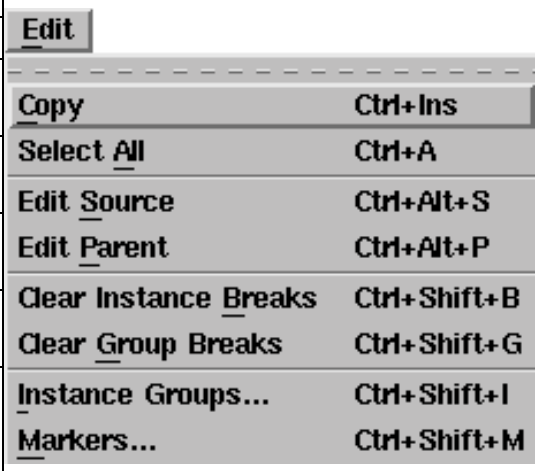
Edit Parent - Opens the source file for the parent scope in a text editor window.

Clear Instance Breaks -Removes all line breakpoints from the instance in use.

Clear Group Breaks - Removes all line breakpoints from the instance group in use.

Instance Groups - Opens the Instance Group Edit Dialog. The dialog is used to create instance groups.

Markers - Opens the Markers Dialog. The dialog is used to define markers for specific simulation times.



Display Menu

Show Execution - Shows execution information for the source file in the Execution pane. Arrows show activity.

Show Coverage - Shows coverage information in place of execution information. Coverage displays the execution counts (number of times the line was executed) and disables stepping through source code.

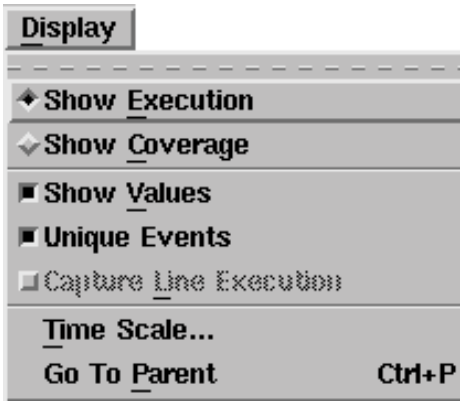
Show Values - Displays current values for all signals. Current signal values appear below the variable name. If there is not enough space for a signal value to appear, an asterisk appears next to the signal name. Moving the cursor over the signal name displays a popup of the current signal value.

Unique Events - Toggles Unique Events *on* or *off*. When *on*, unique events display in the Source Window. When *off*, unique events do not display.

Capture Line Execution - Captures and displays line execution data in Interactive mode.

Time Scale - Opens the Time Scale Dialog, which is used to set the display unit and display precision for VirSim display time.

Go to Parent - Selects the parent of the currently dis9played scope, adds it if necessary to the current instance group, and displays its source in the Source Window.



View Menu

View Menu settings are saved when you close VirSim.

Source Tools - Toggles display of the Source Window tools.

Search Tools - Toggles display of the Search tools.

Window Icons - Toggles display of the Window icons.

Break Group Icon - Toggles display of the Breakpoint Group icon.

Link Icon - Toggles display of the Link icon.

6

Logic Browser

The Logic Browser (Verilog only) is a compact, easy-to-read schematic window designed for analysis of complex designs. Drag a scope, signal or group of selected signals into the Logic Browser and click on ports to expand connectivity in relevant areas. Explore the design behavior by analyzing the annotated values for ports and nets as you go backwards and forwards in time.

The Logic Browser dramatically simplifies debugging of bus collisions, tracing of undesired X values through complex circuitry, and understanding of design connectivity.

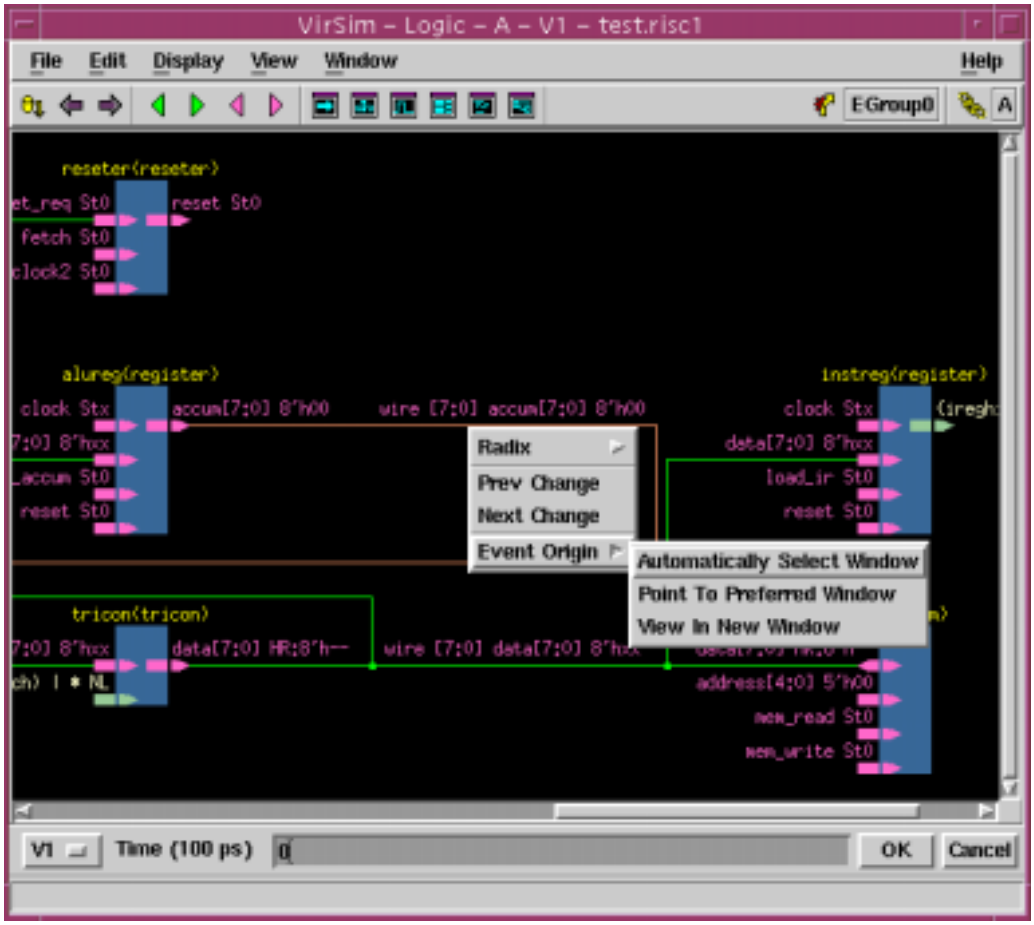
Introducing the Logic Browser

Features

Using Logic Browser you can:

- Easily trace signal connectivity and view change history.
- Perform all functions in post-simulation mode as well as interactive mode. This frees the simulator to perform other work in parallel with debugging.
- View all instances and assignments driving or driven by connected nets.
- See all ports on instances with their names and values.
- Navigate the time domain; jump to the next change in the selected nets OR next change in any displayed net or port.
- Navigate the design domain by simply clicking inside or outside of ports to traverse the hierarchy.
- Conveniently view assignments as instances.
- Find the cause of signal transitions using Event Origin.
- Load values only for signals of interest, saving time and memory.
- Quickly locate signals that changed at the current time

Figure 6-1 Sample Logic Browser



- All instances and assignments driven by the displayed nets are shown.
- The selected nets are displayed in pink, all others are in green.
- Nets are displayed with type, name and current value.
- Ports on all instances are shown with the connection expression text and value of that expression. Note: only values for the connected ports are loaded by default.

Getting Started

The following information is covered in this section:

- Before Using the Logic Browser
- Getting Started

Before Using the Logic Browser

Logic Browser accesses simulation value change results either from a VCD+ file or interactively. The Logic Browser requires connectivity, expressions, variables, and other information that are inappropriate to store in a VCD+ file and are not available from the Verilog PLI. The Logic Browser gets this information from the database image produced by the Verilog Source Compiler (VSC).

When Logic Browser requires connection information, VirSim checks if the Verilog source has been compiled; if not VirSim compiles the source either automatically with the arguments provided on the VirSim command line or manually prompted by the Load Sources Dialog.

The VSC accepts the same command line switches, has the same library search rules, and interprets the design in the same way as the Verilog simulator.

Getting Started

To begin using the Logic Browser, simply drag a scope, signal or selected group of signals into the window from any of the other windows.

Graphic Objects and Value Text

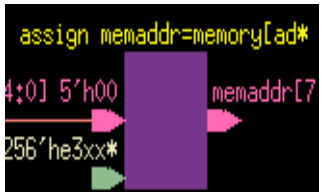
Many graphical objects are used when displaying the loads and drivers of a signal. You may interact directly with the graphic objects to navigate through a design, display signal values, and perform other functions such as editing source files.

The following information is covered in this section:

- Viewing Graphic Objects
- Viewing and Loading Value Text

Viewing Graphic Objects

Object	Text
Port	Value text is displayed to the right of the port name.
Port Instance	Value text is displayed to the right of the port name. Text shows what is connected to the outside of this port (For this example, clock.) When Port and Net is selected from the Display menu, the port definition is displayed. For Verilog, this name is obtained by looking at what is connected on the inside of this port. Note that no definition text is shown when the inside connection is some expression like (i1 | i2).
Primitive Terminal	Value text is displayed to the right of the port name. Text shows what is connected to the outside of this primitive terminal. When Port and Net is selected from the Display menu, the port definition is displayed. For Verilog, this is the terminal position. (For example, the first port is connected to the net clk.)

Object	Text
Module Instance 	Logic Browser displays the module instance name. (For example, SubmodWithBiDirInst) and the type of module instance (for example, Submod).
Primitive Instance 	The primitive instance name. (For example, aPrimitive) is displayed along with the primitive instance type (for example, and).
Assignment Statements 	All contributors to any assignment statement are displayed as drivers to an instance. Assignments act like primitives with the exceptions listed below. an input port appears for each instance of the nets contained in the expression on the left side of the assignment, An output port is created for the net on the right side of the assignment. The high conn expression for each of these ports is the instance of the net that caused the port to be created. Exceptions to the primitive display model are as follows: • The name of the object is the entire assignment statement. • The module definition name does not appear in the logic browser. • There is a single output port. This means that assignments with a concatenation on the left hand side of the equal sign have a single output port with the concatenation expressions as the high conn. • There are no port names. Note: The values displayed on the ports are calculated using the values of the nets at the end of the currenttime. In the case of a procedural assignment, these may not be the actual values that existed when the assignment was executed. In the case of continuous/last assignments with delays, the output value is calculated without analyzing delay information.

Viewing and Loading Value Text

Value text on the Logic Browser display shows variable data such as signal load status, data availability, and data values. It is dynamically updated according to time viewed, signal load status, and display radix selection. The underlying signal values are obtained either from a VCD+ history file (post-processing mode) or from a running simulation (interactive mode).

Any text that is too long to fit in the available space is abbreviated and followed by an asterisk. Place your cursor over the abbreviated text and the full text appears in the status bar. Allowable length for value text is controlled with an X resource. See Chapter 16, VirSim Setup.

Loading Signal Values

For best performance, the Logic Browser supports incremental signal loading. An NL indicates that the signal has not yet been loaded for value display. After a load, the NL value text is replaced by actual simulation values. A connection with NL value text may require loading of multiple signals, depending on the expression. To avoid unnecessary load delays, use the toolbar Load Value Changes icon and the Logic Browser CSM Load Values commands judiciously. Because a Logic Browser view may contain hundreds of signals, a substantial delay may occur when these commands are used with extremely large files. To avoid lengthy delays, use the appropriate context sensitive menus to load only signals for specific modules or port instances.

Note:

To automatically load values for all objects shown in the Logic Browser, select Autoload Values from the Display menu.

To	Do This
Load all signal values for the current view.	Click the Load Signal Values icon or select Load View NL from the Logic Browser context sensitive menu.
Load values for all signals connected to ports on a specific module instance.	Select Load Inst NL from the Module Instance context sensitive menu.
Load values for one or more signals required to display the value associated with a port instance.	Select Load Values for the Port Instance context sensitive menu. Often only a single signal is loaded, but multiple signals may be loaded in the case of complex expressions.

Unavailable Signals

A value of NA means that the signal is not available as it was not saved in the database. History recording for scopes and signals may be turned on or off at any time during a simulation.

Hierarchically Resolved Signals

When displaying values for a net, an HR in front of the value indicates that the value of the signal at the port was hierarchically resolved. For a port this means this is the combined value of all the drivers outside of the current hierarchy.

For a port instance, this is the combined value of all the drivers inside the instance. For a primitive terminal this is the value that terminal is driving onto the signal.

Note that HR values only occur when a signal has more than one driver. A port value - means that VirSim only has the resulting net value and not the values of individual drivers. Add `+vpddrivers` to the simulator command line to save this data.

Note:

When a net is driven by multiple instances, the net shows the resultant value, while each driving port shows a hierarchically resolved (HR) value. The HR value is the net contribution from within this instance. For example, if the net resultant value should be 0 but is X and the two driving ports show HR:0 and HR:1, then we know to debug the 1 path. An open wire without fanins may still be an input to control statements such as @ (posedge clock), if (cnt > 0) etc. Drag the wire into a SW and use the Find arrows to locate its uses. When examining output port values from assignments, be aware that they reflect the current value of the driven net and not the current or scheduled value of the expression. Control statements and delays will need to be considered to determine if that assignment has affected the net value.

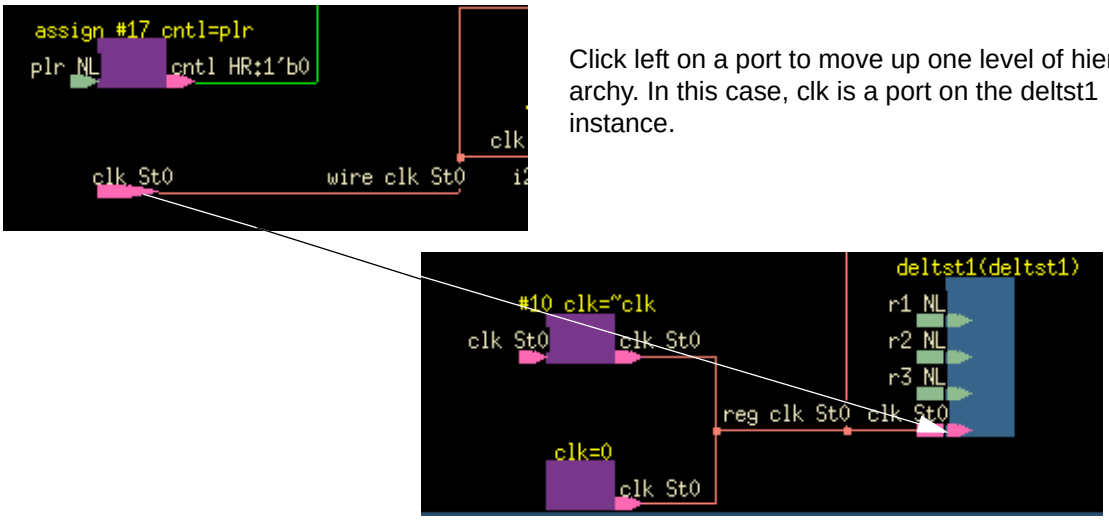
Navigating a Design

This section includes the following information:

- Navigating the Design in the Time Domain
- Navigating Using the Connection Dialog
- Navigating the Design with Graphic Objects

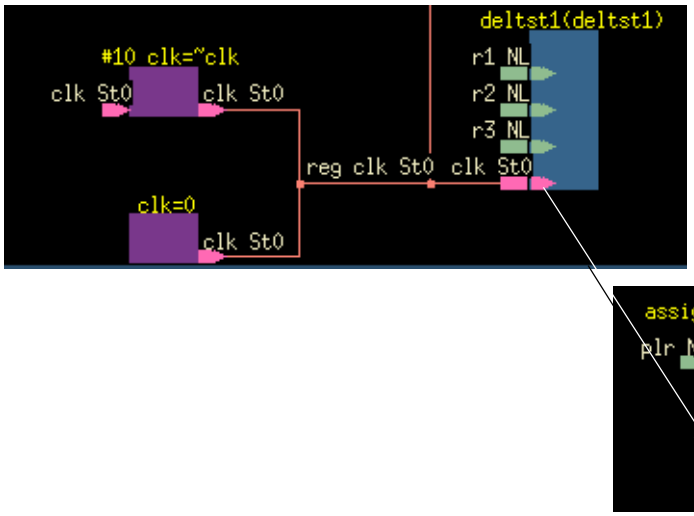
Navigating the Design with Graphic Objects

To move up one level of the hierarchy:



Click left on a port to move up one level of hierarchy. In this case, clk is a port on the deltst1 instance.

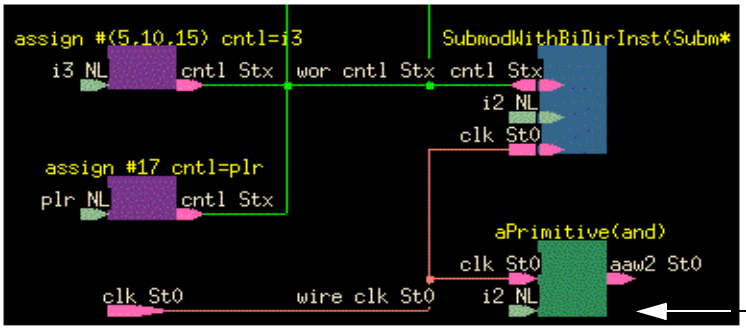
To descend one level of the hierarchy:



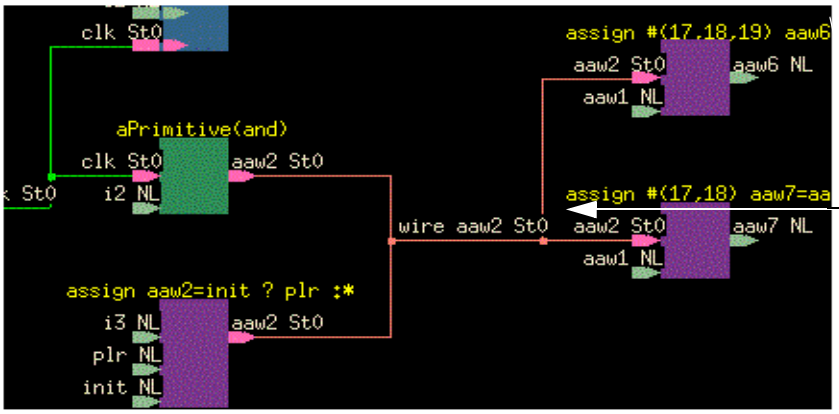
Click left on the inside of a port to navigate to a lower level of the design hierarchy. Signal routing from your previous view is maintained.

To clear signal routing, select Clear All Routing or Clear Displayed Routing from the Edit menu.

To navigate across the design hierarchy:



Click left on the outside of a port to add a wire to the schematic



The new, or selected, wire appears in pink. All others appear in green.

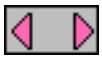
Navigating the Design in the Time Domain

Using Next/Previous Net Change Commands

To navigate the time domain using Next/Previous, either click:

Green arrows

Next/Previous Selected Signals Change icons for the next/previous change in the selected net.



Pink arrows

Next/Previous Any Change icon for the next/previous change in any displayed net that has values loaded.

You can also click right on an object to display an associated context sensitive menu. Choose the Next/Prev Change commands to move the view backward and forward in time. The values affected depend on the type of object selected and Verilog expressions for that object. For example, in the Port Instance CSM, Next Change advances the view to the next time the value changes for the port instance expression. In the Module Instance CSM, Next Change takes into account all value expressions on the periphery of that instance and advances the view to the nearest time change of any one of these expressions.

Using the Time Field

Enter a valid time in the Time field at the bottom of the Logic Browser (see Figure 6-2, Time Field). The Time field rejects non-numeric values and time values that are out of range for the loaded VCD+ file. Then press [ENTER] or click left on OK.

Figure 6-2 Time Field

Using Window Links

Change the time in a window to which the Logic Browser is linked. Logic Browser updates the values for that time.

Navigating Using the Connection Dialog

The Connection Dialog can be useful when navigating complex expressions.Click on an object and select View Connection from the context sensitive menu. The Connection Dialog displays a detailed port connection breakout map, Double click on a navigable signal or bit from one of the lists. Logic Browser displays the selected signal.

An item is not navigable if it is generated from another expression, such as a single bit generated from the expression a & b or a nonexistent bit due to a port size mismatch. For more information, see “Using the Connection Dialog” on page 6[ChapTitle]>6-14.

Using Event Origin

Using the position of the mouse pointer as a reference, the Event Origin feature can obtain precise information about the source of a value change at a specific time. For more information, see Chapter 12, Event Origin.

To	Do This
Find the source of a value change on the net.	Position the cursor over a net and click right to open the context sensitive menu. Select Event Origin.
Find the source of a value change on a connected signal.	Point and click right on the port instance symbol (either the HiConn or LoConn half) to open the Port Instance context sensitive menu and select Event Origin. For primitive terminals and port instances, Event Origin is done on the high conn net. For ports, Event Origin is don on the low conn net. Event Origin is not available if the port is connected to an expression.

Using the Connection Dialog

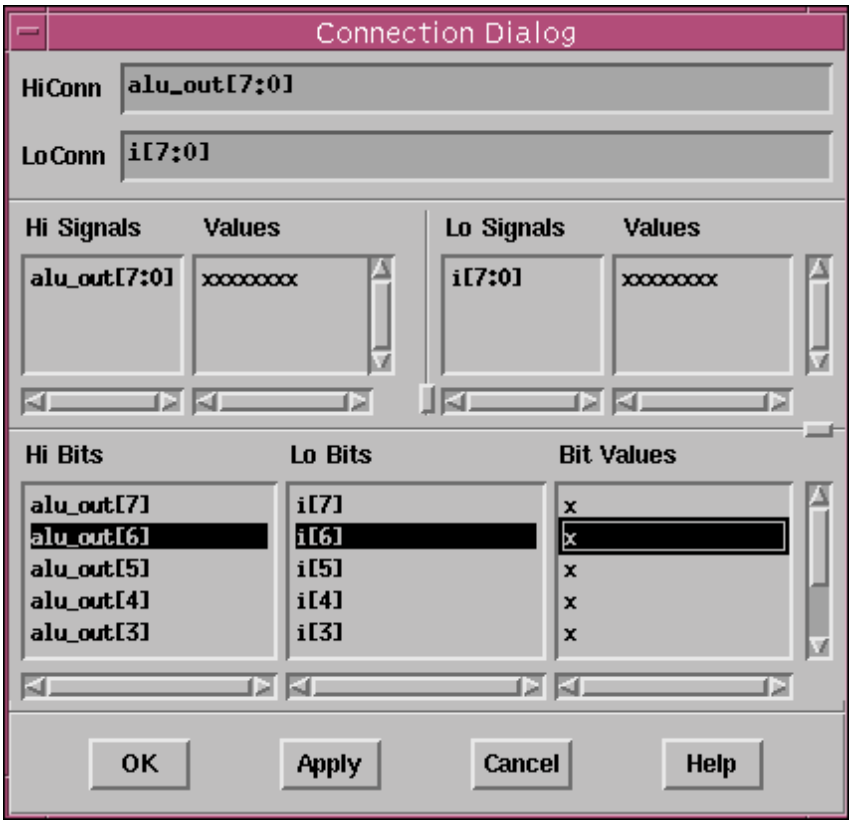
The Connection Dialog is a complex dialog that contains two text fields and seven scroll lists (see Figure 6-3, Connection Dialog). The Connection Dialog displays a detailed port connection breakout map, which can greatly aid in debugging connectivity errors. To open the Connection Dialog, click on an object and select View Connection from the context sensitive menu.

The following information is covered in this section:

- Navigate from the Connection Dialog
- Select a Signal with Expressions

- Expressions Example

Figure 6-3 Connection Dialog



Navigating from the Connection Dialog

To navigate directly from the Connection Dialog, either:

- Double click left on navigable signals or bits from one of the lists.
- Select a navigable signal or bit and click left on the **OK** or **Apply** button.

An item is not navigable if it is generated from another expression, such as a single bit generated from the expression a & b or a nonexistent bit due to a port size mismatch.

Selecting a Signal with Expressions

Verilog permits complex expressions to drive port and terminal connections. A basic problem with complex expressions is that they can present ambiguity in signal selection. The Connection Dialog lets you further define the focus.

The HiConn expression and LoConn expression fields display the text of the HiConn and LoConn port expressions for a port instance. (Because primitive terminals do not have a LoConn, their LoConn field is empty.)

The HiConn Signal/Value and LoConn Signal/Value lists show all of the signals used in the port connection. For simple port connections, there may be only a single signal/value pair. For more complex port connections, the list displays each signal of the port expression and that signal's value.

The HiConn/LoConn Bit and Signal/Value lists can be very useful for tracking connectivity problems, because they display a bit per bit, right justified text, and mapping of the HiConn port expression to the LoConn port expression. The list displays the least significant to most significant bits from top to bottom. The string "***" is used as a place holder for constants and parameters.

The HiConn Signal/Value, LoConn Signal/Value, and HiConn/ LoConn Bit/Value lists scroll in synchronization as a group.

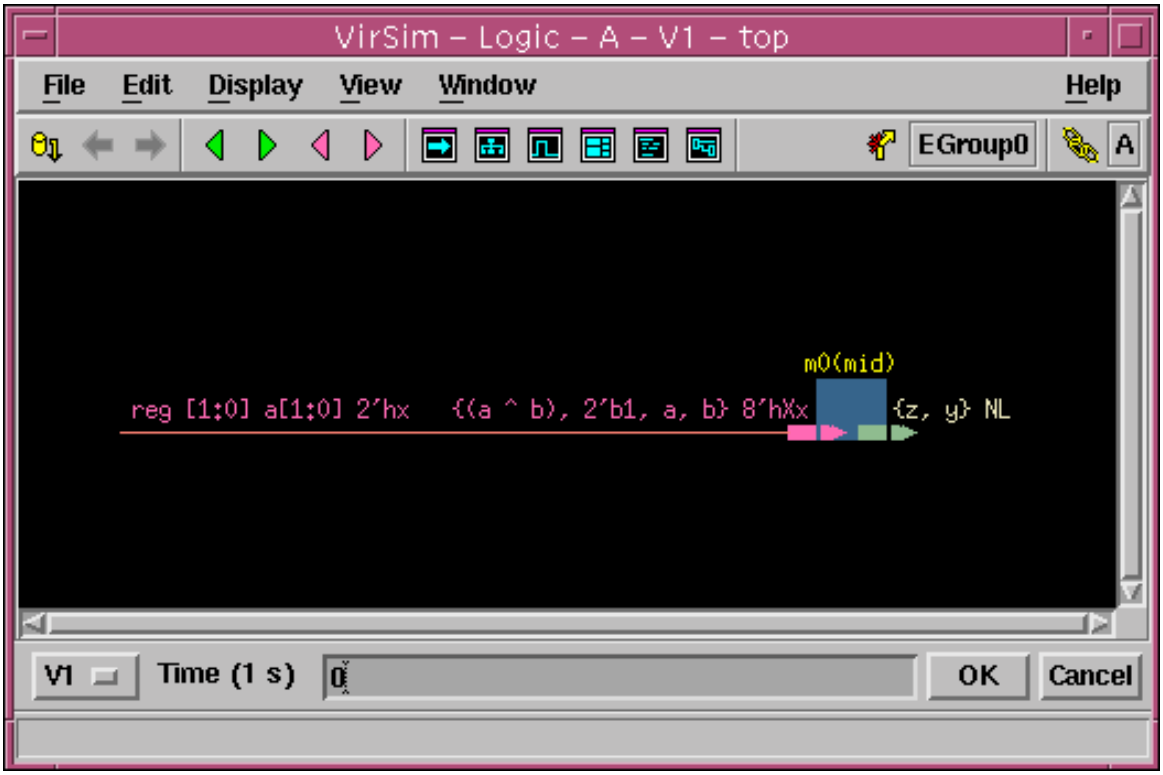
Expressions Example

The following example illustrates the types of complex port expressions that are legal in the Verilog HDL. Consider the following Verilog code:

```
module mid ({i1,i2}, {o1, o2[3:2], o2[0]});
             input  [4:0] i1;
    input  [2:0] i2;
    output       o1;
    output [3:0] o2;
    buf g0 (o2[3], i1[4]);
    not g1 (o2[2], i1[3]);
    buf g2 (o2[1], i1[2]);
    not g3 (o2[0], i1[0]);
    buf g4 (o1, &i2);
endmodule
module top;
wire       y, z;
wire [1:0] netv1;
reg  [1:0] a, b;
initial
begin
    $vcdplusdeltacycleon;
    $vcdpluson;
    $vcdplustraceon;
    repeat (10) #10 {a,b} = $random;
end
mid m0 ({a ^ b, 2'b01, a, b}, {z,y});
and andNet2 (z, y, a[0]);
assign netv1 = {y | a[0], y & a[0]};
endmodule
```

Assume that we have compiled this design and we view the focus *top.a* in the Logic Browser. The Logic Browser view is shown in Figure 6-4, Connection Dialog Example.

Figure 6-4 Connection Dialog Example



Note the port connection expressions {a^b,2'b01,a,b} and {z,y}. Right click on the port instance with expression text {a^b,2'b01,a,b}, and choose View Connection from the context sensitive menu. The Connection Dialog appears (see Figure 6-3, Connection Dialog). By double-clicking on either i1 or i2 you can select which signal to follow when descending the hierarchy into the mid module.

Editing Source from Logic Browser

The Logic Browser includes two commands for opening source code in a text editor: Edit at Definition and Edit at Instance. The editor used to edit text is determined from the EDITOR environment variable. In Unix, if there is no editor environment variable set, the editor defaults to VI. In Windows, if there is no editor environment variable set, the program looks for a file association. If there is no file association, the editor defaults to WordPad.

To	Do This
Open the Source file for the current instance.	Right click on a Module instance and select Edit Current Instance for the Module Instance Context Sensitive menu.
Open the Source file for the module type definition	Right click in the Logic Browser graphics area to open the Logic Browser context sensitive menu, or right click on a module instance to bring up the Module Instance context sensitive menu. From either menu, select Edit Module Type Definition.

Note:

If you change the source file you need to resimulate, reload data and recompile the source in VirSim.

Changing the Display

Port and Net Display Setting

In the menu bar, the Display menu includes two commands for toggling the display text settings. Choose Net Only to display net names only. Choose Port and Net to display both port and net names.

Time Scale Settings

You can select the display unit and display precision for time values from the Time Scale Dialog. To open the Time Scale Dialog, click left on the Display menu in the menu bar and choose Time Scale. A dialog provides the time-scale selections. For more information on the Time Scale Dialog, see Chapter 10, Time Units.

Show Single or Multiple Nets

In most cases you will want to use the Logic Browser's multi-net schematic. Choose Show Single Net from the Display menu to view a simpler, single-net at a time view. This corresponds to the pre-VirSim 4.0 Logic Browser.

Note:

We recommend that you do not switch back and forth between single and multi-level views. The two modes share a data structure, and some history may be lost.

Displaying Unique Events

From the Display menu, toggle Unique Events on or off. When on, unique events display in the Logic Browser.

Change the Radix Display

Right click on either the selected net or the port instance and select Radix from a context sensitive menu. From the displayed Radix menu, you can select a radix base (binary, hexadecimal, octal, etc.) for the selected port instance or net. Refer to Chapter 11, Radices.

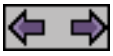
Changing the Color Scheme

Refer to Logic Browser Settings on page 16-12.

Toolbar and Menu Reference

Toolbar

You can toggle display of toolbar icons from the View menu.

Load Value Changes	The Load Value Changes icon loads values for all signals in the current view. After a load, the NL value text is replaced by NL value text may require loading of multiple signals, depending on the expression. This command is sensitive only if there are NL values in the view. To avoid lengthy load delays, use the toolbar icon judiciously. Because a Logic Browser may contain hundreds of signals, a substantial delay may occur when this command is used with extremely large files. To avoid lengthy delays, use the appropriate context sensitive menus to load only signals for specific modules or port instances. Following a load operation, the affected Logic Browsers automatically refresh for new data (even when the load is triggered from another window). NL values immediately update to show actual simulation values.
 **Previous View Next View**	These icons are used to move through previous Logic Browser views. In multiple-net mode, if you add nets to a view, the added nets will appear when you move to the scope using the Previous/Next View tools. VirSim always navigates according to the current Single/Multiple level setting in the Display menu. For example, if you are in single-net mode and use the Previous View button to view a multi-net view, any navigation from the multi-net view will be in single net mode. We recommend that you do not switch back and forth between single and multi-level views because the two modes share a data structure, and some history may be lost.

Next Selected Signals Change	The Next Selected Signals Change icons search forward or backward in time for next or previous value change in selected signals. When a change is located, all values are updated to reflect the new time, and the value change is highlighted to indicted which values have changed. Any other windows that are linked also update to the new time.
Next Any Change	The Any Change icons work similarly to the Selected Signals Change icon except that all signals in the view are used to find the next value change. Consequently, the time changes affected by these arrows are always less than or equal to those affected with the Selected Signals Change Arrows. If you want to find the next or previous change for an individual object or instance, use the context-sensitive menu.
New Window Icons	These icons allow you to open new VirSim windows from the Logic Browser toolbar.
Breakpoint Group Icon	The Breakpoint Group icon opens the breakpoint groups menu. Breakpoint groups consist of groups of expressions that are used to search for breakpoints. You can select available breakpoint groups from the menu. Breakpoint groups are created in he Breakpoint Groups Dialog. The group expressions are defined in the Expressions Dialog. For information on creating Breakpoint Groups, see Locating Events Using Breakpoint Groups with Enabled Expressions.
Link Icon	The Link icon links windows so that any change in time in one window is reflected in all other windows. A capital letter in the Link buttons shows the current link. To link or unlink a window, click the Link button and select the desired link.

Context-Sensitive Menus (CSM's)

Logic Browser context-sensitive menus provide convenient access to the most frequently used operations for various types of objects. The CSM opens when you point and click right on an object.

The following context-sensitive menus are covered in this section:

- Logic Browser CSM
- Module Instance CSM
- Module Instance CSM
- Port Instance CSM
- Selected Net CSM

Note:

The editor used to edit text is determined from the EDITOR environment variable. In NT Windows, VirSim can also use an editor associated with the file in the Windows File Manager. For information on associating files, see Windows NT documentation.

Logic Browser CSM

Commands in the Logic Browser CSM apply to the whole Logic Browser view (see Figure 6-5, Logic Browser CSM). To open the Logic Browser CSM, point and click right in the Logic Browser graphics area (in free-space), but not over an object.

Figure 6-5 Logic Browser CSM

Prev/Next Change
Searches backward and forward in time for previous or next value changes on any object in the view for which values are loaded. These menu commands are sensitive whenever one or more port instances are loaded.

Load View NL
Loads all signals in the current view. After a load, the NL value text is replaced by actual simulation values. A connection with NL value text may require loading multiple signals, depending on the expression. This command is sensitive only if there are NL values in the view.

This command is equivalent to using the Load button in the toolbar and similar precautions apply. Because a Logic Browser view may contain hundreds of signals, a substantial delay may occur when this command is used with extremely large files.

Note:
To avoid lengthy delays, use other context sensitive popup menus to load only signal values for specific modules or port instances.

Edit At Definition
Opens an editor at the first line of the module type definition for the scope which contains the selected signal. For example, if the current signal's are in the scope top.cpu.regpc, then the source of the module type definition for type reg32 is displayed in the editor.

Note:

The editor used to edit text is determined from the EDITOR environment variable. In Unix, if there is no editor environment variable set, the editor defaults to VI. In Windows, if there is no editor environment variable set, the program looks for a file association. If there is no file association, the editor defaults to WordPad.

Module Instance CSM

The commands in the Module Instance CSM apply only to that module instance and do not affect other Logic Browsers that may display the same module instance (see Figure 6-6, Module Instance CSM). To open the Module Instance CSM, point and click right on a module instance symbol.

Figure 6-6 Module Instance CSM

Prev/Next Change
Searches backward and forward in time for any changes in the periphery of specific module instances. On any object for which values are loaded (not NL), the view moves forward or backward to the previous or next value change. These menu commands are sensitive whenever one or more port instances are loaded.

Load Inst NL
Loads values for all signals connected to ports on this module instance. The command is sensitive only if the instance has ports that have not been loaded (NL).

Edit At Definition
Opens an editor at the module type definition line for the module instance.

Edit At Instance
Opens an editor for the file containing the definition of the module type for the instance. If possible, you are positioned at the line where the module definition starts (the “module” line).

Note:
The editor used to edit text is determined from the EDITOR environment variable. In Unix, if there is no editor environment variable set, the editor defaults to VI. In Windows, if there is no editor environment variable set, the program looks for a file association. If there is no file association, the editor defaults to WordPad.

Port Instance CSM

The commands in the Port Instance CSM apply to that port instance only and do not affect other Logic Browsers that may display the same port instance (see Figure 6-7, Port Instance CSM). To open the CSM, point and click right on the port instance symbol (either the HiConn or LoConn half).

Figure 6-7 Port Instance CSM

Radix
Opens the Radix menu. Selecting a radix sets the display value for that port instance only to the selected radix base (binary, hexadecimal, octal, etc.). Refer to Chapter 11, Radices.

Prev/Next Change
Searches forward or backward in time to the next or previous value change for the selected port instance. Value changes show expression values at different points in time for the port instance. These menu commands are sensitive only when all signals are loaded.

Event Origin
Obtains precise information about the origin of the current value of the connected signal. Refer to Chapter 12, Event Origin. For primitive terminals and port instances, the high conn net is used. For ports, the low conn net is used. This option is not available if the port is connected to an expression.

Load Value
When selected, loads one or more signals required to display the value associated with the port instance. Often, only a single signal is loaded, but multiple signals may be loaded in the case of complex expressions. This command is sensitive only if the port value has not been loaded (NL).

View Connection
Opens the Connection Dialog (see Using the Connection Dialog on page 6-14) and displays connection information for the port instance. The Connection Dialog remains open across multiple View Connection operations on different ports, port instances, and primitive terminals.

Selected Net CSM

The commands in the Selected Net CSM apply to the net on which your cursor is pointing and do not affect other Logic Browsers that may display the same net (see Figure 6-8, Focus CSM). To open the CSM, position the cursor over a net and click right.

Figure 6-8 Focus CSM

Radix
Opens the Radix menu. Selecting a radix sets the display value for that focus net only to the selected radix base (binary, hexadecimal, octal, etc.). Refer to Chapter 11, Radices.

Prev/Next Change
Searches forward or backward in time to the next or previous value change for the selected net. Value changes show expression values at different points in time for the net. These menu commands are sensitive only when all signals are loaded.

Event Origin
Obtains precise information about the origin of the current value of the net. Refer to Chapter 12, Event Origin.

Menu Bar

Note:

Any menu with a dotted line at the top can be torn off and placed on the display for easy access. Click middle above the dotted line, and then drag the menu to a convenient position.

File Menu

Open - Opens the file browser to load an EPIC, VCD, or VCD+ history file.

Reopen - Reopens all history files that have changed since they last were loaded.

Close - Closes an open history file.

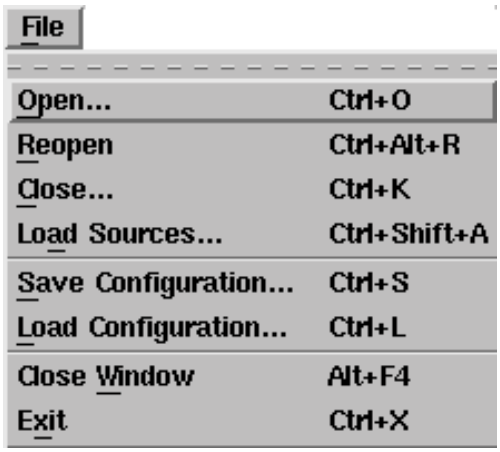
Load Sources - Compiles Verilog source.

Save Configuration - Saves the current configuration.

Load Configuration - Loads a configuration file.

Close Window - Closes the Logic Browser window.

Exit - Exits VirSim.

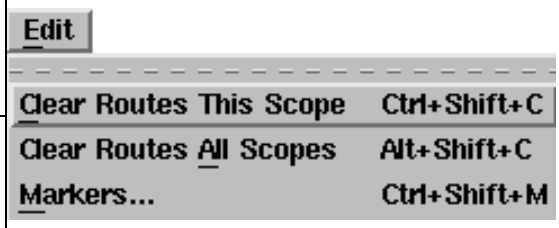


Edit Menu

Clear Displayed Routes - Clear the displayed routes. To view routes previous to the last change, drag the signal back into the Logic Browser from another window.

Clear All Routes - Clear all routes, even those not currently displayed. Dragging a signal in again displays the signal with no previous routing.

Markers - Open the Markers Dialog. The dialog is used to define markers for specific simulation times.



Display Menu

Show Multiple Nets - Displays a multi-net schematic,

Show Single Net - Displays a single net at a time, which corresponds to the Focus Net mode in previous VirSim Releases.

Net Only - Displays only the net name at the port instance.

Port and Net - Displays the net and port name at the port instance.

Time Scale - Open the Time Scale Dialog for setting the display unit and display precision for VirSim display time.

Unique Events -Toggle Unique Events *on* or *off*. When *on*, unique events are displayed with signal values in the Logic Browser. When *off*, unique events do not display.

Autoload Values - Automatically loads signal values as new signals are added to the display.

Note:

We recommend that you do not switch back and forth between single and multi-level views. The two modes share a data structure, and some history may be lost.

View Menu

View Menu settings are saved when you close VirSim.

Source Browser Tools - Toggles display of the Logic Browser tools.

Search Tools - Toggles display of the Search tools.

Window Icons - Toggles display of the Window icons.

Break Group Icon - Toggles display of the Breakpoint Group icon.

Link Icon - Toggles display of the Link icon.

7

Register Window

The Register Window is used to create Design Views using simple graphics tools. The view can show items such as boxes, lines, text, signal values and expression results at specific simulation times, signal names, and user created labels.

Introducing the Register Window

Window Areas

Note:

If you use a two-button mouse, you cannot drag and drop signals from the Register Window to other windows. Instead, you must cut or copy and paste.

The Register Window consists of one main display area, a Time field, and a status bar. These are described below.

Display Area	The design view is created and displayed by placing signals and objects in the display area. You can load the desired design view from a previous configuration or construct it within the display area. The Register Window provides basic drawing tools to construct and enhance the view.
Time Field (Time Control Text Field)	Use the time field to jump to a specific time. To display values for a new simulation time, enter the time and press Enter or click left on the **OK** button. All windows linked to the Register Window also display the new simulation time. The time entered must also be within the time range of the simulation file.
Status Bar	The Status Bar, located at the bottom of the window, displays descriptions of window areas, hierarchical signal names, descriptions of unique event indicators, and control buttons within the window.

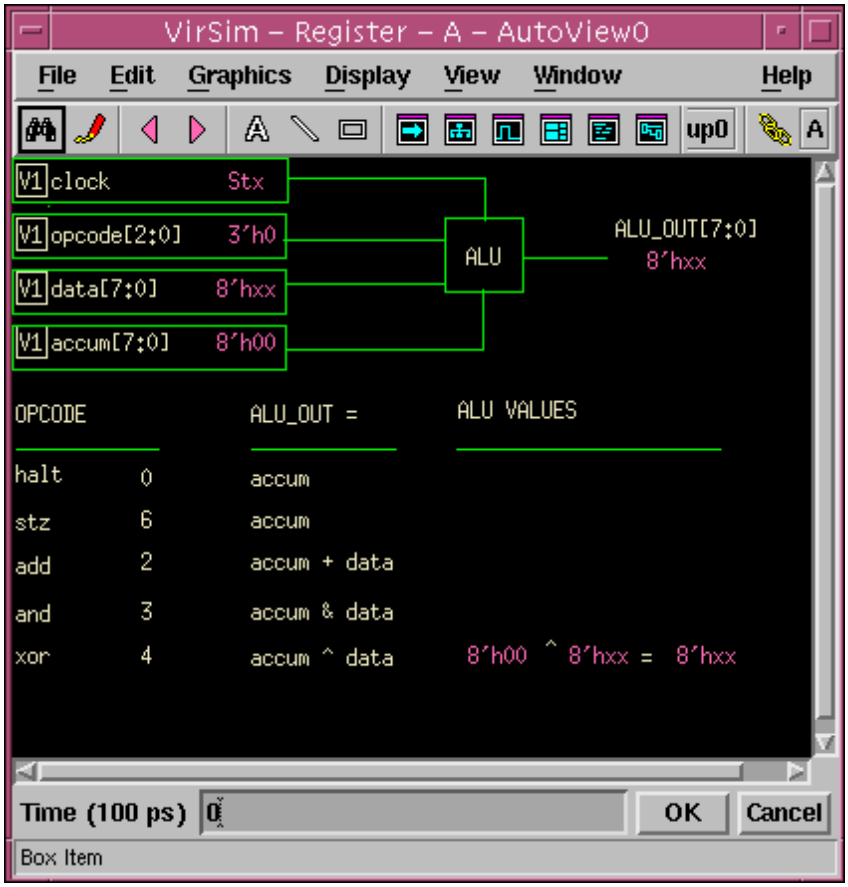
What is a View?

A view is a free graphics display that is often used as a scratch pad. It can be saved to the configuration file and reused by loading the configuration file. A name is assigned to the view with View Editor (opened from the Edit menu). To display a menu of available views, click left on the binoculars icon in the toolbar. Automatically named views are called AutoViewN, where N is a sequentially assigned number starting with 0 for the first group.

Which Signals are Loaded?

By default, all signals for all defined Register Window views and Waveform Window signal groups are registered in interactive mode or loaded in post-processing mode, regardless of whether they are currently being displayed. You can modify VirSim's setup to register or load only those signals that are displayed in the currently active Waveform signal group or Register Window view. This may be more efficient when you are analyzing large designs. For more information, see Common Settings on page 16-4.

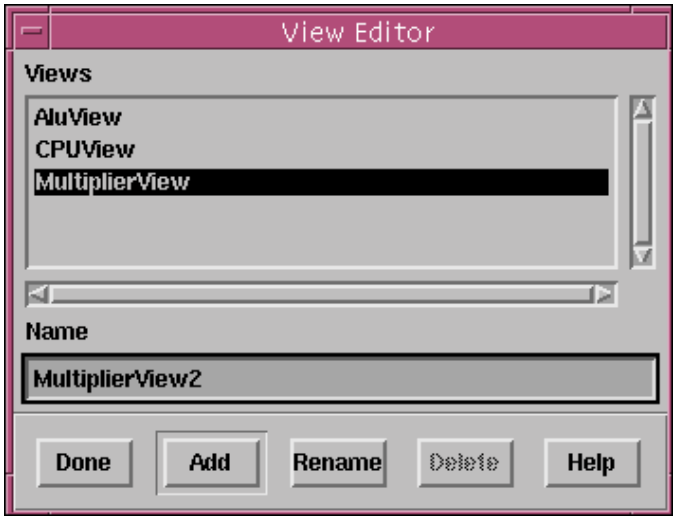
Figure 7-1 Sample Register Window



Edit Views

The View Editor Dialog is used to create, display, rename, and delete views (see Figure 7-2, View Editor Dialog). A view is a named collection of text, signals, and graphics displayed in the Register Window.

Figure 7-2 View Editor Dialog



In addition to views manually created with the View Editor Dialog, a view is automatically created when anything is added to an empty Register Window.

The View Editor Dialog has the following display areas:

Views

Displays a list of the currently defined views.

Name

Used to enter or edit a view name. Names may only contain alphanumeric characters, underscores, and dollar signs. Names must begin with an underscore or alpha character.

To	Do This
Create a View	Enter the view name in the Name field and click left on the Add button.
Delete a View	Click left on the view name in the Views list and click left on the Delete button.
Rename a View	Click left on the view name in the Views list. Then click left in the Name field and enter the new name. The arrow keys and backspace key may be used to edit the name. After modifying the name, click left on the Rename button.

Undo Command

You can undo the effects of most commands using the Undo command. To undo the last command, click left on the Edit menu and choose Undo.

Using Event Origin

Event Origin can obtain precise information about the cause of a signal value change. Point and click right on a signal to open a context sensitive menu and select an option from the Event Origin menu. For more information, see Chapter 12, Event Origin.

Display Unique Events

(Verilog only) The Unique Events feature allows you to display symbols on signal values for events of interest in the Waveform Window, Register Window, Source Window, and Logic Browser. To display unique events click left on the Display menu in the menu bar and select Unique Events. For more information on creating Unique Events, refer to Chapter 4, Waveform Window.

Set Time Scale

You can select the display unit and display precision for time values from the Time Scale Dialog. To open the Time Scale Dialog, click left on the Display menu in the menu bar and choose Time Scale. A dialog provides the time-scale selections. For more information on the Time Scale Dialog, see Chapter 10, Time Units.

Align Objects

You can select and align objects (left, right, top, bottom, vertical center, or horizontal center) in relation to an anchor object. The last object you select becomes the anchor object. To align objects:

1. Select the objects you want to align.
2. Click left on the Graphics menu and choose Align.
3. From the Align popup menu, choose the desired alignment: Left, Right, Top, Bottom, Vertical, or Horizontal.

Distribute Objects

You can select and distribute objects equidistant between two points. The two points are determined to be the two extreme points of the selected objects (see Figure 7-3, Distributing Objects Example).

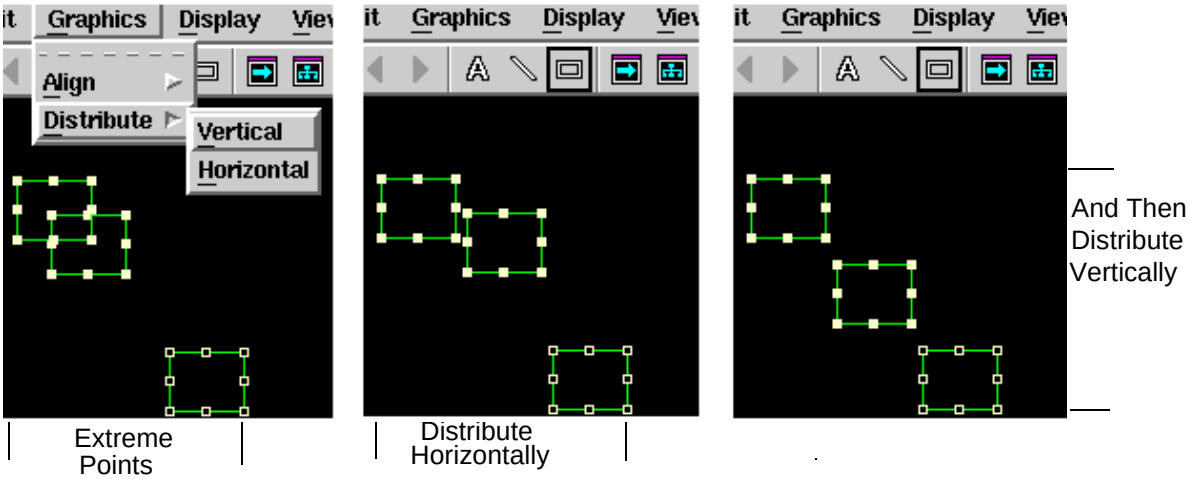
To distribute objects:

1. Select the objects you want to distribute.
2. Click left on the Graphics menu and choose Distribute.
3. From the Distribute popup menu, choose the desired distribution: Vertical or Horizontal.

Display File Designators for SIgnals

The Display menu Designator command toggles file designators for signals *on* or *off*. When turned *on*, the file designator appears in front of the signal (for example, V1 clock), so that when comparing data from more than one file, you know which file the signal comes from. When turned *off*, only the signal name appears (for example, clock). To toggle the file designator *on* or *off*, in the menu bar click left on Display and choose Designator.

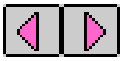
Figure 7-3 Distributing Objects Example



Toolbar and Menu Reference

Toolbar

You can toggle display of toolbar icons from the View menu.

 **Previous/Next Change Icons**	The Previous/Next Change icons search backward and forward in time for the next or previous change on any of the signals in the current view. It then updates all values in the Register Window for the new time. The signals that changed value since the previous display are presented in pink, while the unchanged signals are white. All linked windows will follow and display the state at this new time.
View Icon	The View icon opens a pop-up list of all defined views. To display another view in the Register Window, click left on the View icon in the toolbar to open the pop-up list and then click left on the desired view.
Marker Icon	The Marker icon opens a pop-up list of all defined markers. Click left on a marker to set the Register Window and all windows linked to the Register Window to the simulation time of that marker. For information on defining markers, see Chapter 14, Markers.

 Text Tool Icon	The Text Tool icon activates a text cursor to enter text in the register display area. Click left on the Text Tool icon and move the cursor anywhere you want in the register display area. then click left to position the cursor at the insertion point and type the text. • To delete the previous character(s), press the Backspace key. • To start a new line, press Return. • To cancel text mode, press Esc (escape key) or select another function. • To edit text, select the Text Tool and click left in existing text.
 Line Tool Icon	The Line Tool icon is used to draw straight lines. Click left on the Line Tool icon to select this drawing mode. Depress the left mouse button where the line is to start. Hold down the left mouse button and drag the line to the end point. Release the button to complete the line. Simultaneously pressing Shift while drawing the line forces the line to snap to grid. To edit a line, select the line, hold down the left mouse button on an end point, and drag it to a new location. Release the mouse button to complete the move.
 Rectangle Tool Icon	The Rectangle Tool icon is used to draw rectangles. Click left on the Rectangle Tool to select this drawing mode. Depress the left mouse button where the first corner is to be placed. Hold down the left mouse button where the first corner is to be placed. Hold down the left mouse button and drag diagonally to form the desired rectangle. Release the button to complete the rectangle. To edit a box, select the box, hold down the left mouse button on any edit point, and drag the box to a new location. Release the mouse button to complete the move.
 New Window Icons	These icons allow you to open new VirSim windows from the Register Window toolbar.

Breakpoint Group Icon	The Breakpoint Group icon opens a pop-up list of available breakpoint groups. Breakpoint groups consist of groups of user-defined HDL expressions that may be enabled for searching. You can select available breakpoint groups from the menu. Breakpoint groups are created int he Breakpoint Groups Dialog. The expressions are created in the Expressions dialog. For information on creating breakpoint groups, see Locating Events Using Breakpoint Groups with Enabled Expressions.
Link Icon	The Link icon links windows so that any change in time in one linked window is reflected in all linked windows. The capital letter next to the Link icon shows the current link. To link or unlink a window, click left on the Link icon and choose a different link.

Signal Properties CSM

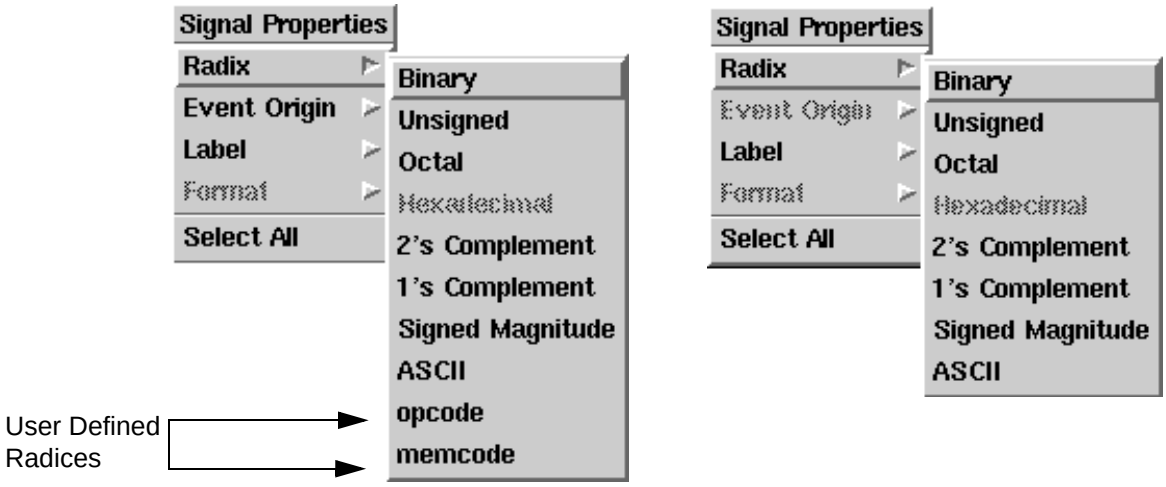
The Signal Properties CSM is used to change radix, control signal name display, control right-left justification, and find the cause of signal value changes (through the Event Origin command). To open the CSM, point and click right on a signal. This opens the Signal Properties menu (see Figure 7-4, Signal Properties CSM).

Figure 7-4 Signal Properties CSM

The Signal Properties menu contains the following selections:

Radix
The Radix selection is used to change the radix of a signal value. To change the radix, click left on Radix and choose a new radix from the list of standard and user-defined radices (see Figure 7-5, Radix Menu). The available choices depend on whether the signal is a scalar or vector.

Figure 7-5 Radix Menu



Verilog Radix Menu **VHDL Radix Menu**

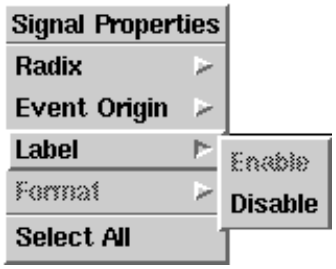
Event Origin

(Verilog only) Event Origin can obtain precise information about the source of a signal value change in the Register Window at a specific time. For more information, see Chapter 12, Event Origin.

Label

Label is used to enable or disable the display of the signal name for selected signals. Click left on a label and choose Enable or Disable (see Figure 7-6, Label Selection Options).

Figure 7-6 Label Selection Options



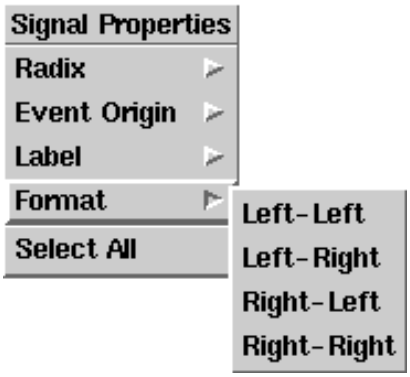
Format
Format is used to format the text position of a selected group of signals, (see Figure 7-7, Format Text Positioning Options). Click right on text to open the Format menu and then choose from the following left and right text positions: Left-Left: Lines up the left side of each selected label and value text pairs.

Left-Right: Lines up the left side of each label text and the right side of each corresponding value text of each selected pair.

Right-Left: Lines up the right side of each label text and the left side of each corresponding value text of each selected pair.

Right-Right: Lines up the right side of each selected label and value text pair.

Figure 7-7 Format Text Positioning Options



Select All
Selects all signals and objects in the window.

Menu Bar

The following menu bars include commands specific to the Register Window:

- File Menu
- Edit Menu
- Graphics Menu
- Display Menu
- View Menu

Note:

Any menu with a dotted line at the top can be torn off and placed on the display for easy access. Click middle above the dotted line, and then drag the menu to a convenient position.

File Menu

Open - Displays the file browser to open an EPIC, VCD, or VCD+ history file.

Reopen - Reopens all history files that have changed since they last were opened.

Close - Displays the Close File dialog to close an open history file.

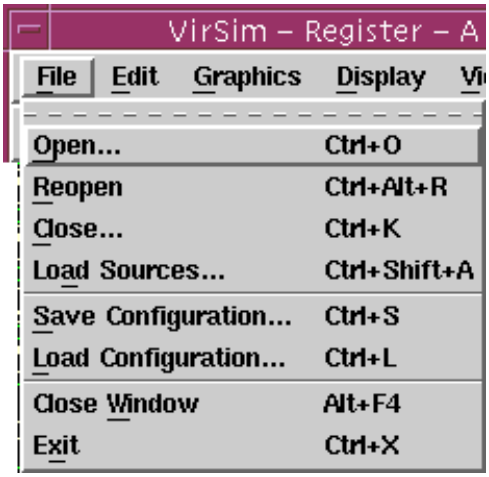
Load Sources - Displays the Load Sources dialog to compile Verilog source.

Save Configuration - Displays the Configuration dialog to save the current configuration.

Load Configuration - Displays the Configuration dialog to load a configuration file.

Close Window - Closes the Register Window.

Exit - Exits VirSim.



Edit Menu

Undo - Undoes the effects of your last command.

Cut - Cuts the selection and moves it to the clipboard.

Copy - Copies the selection to the clipboard.

Paste - Pastes the clipboard to a designated area.

Delete - Deletes the selection.

Select All - Selects all signals and objects in the window.

Views - Opens the **Views** Dialog to create, rename, and delete views.

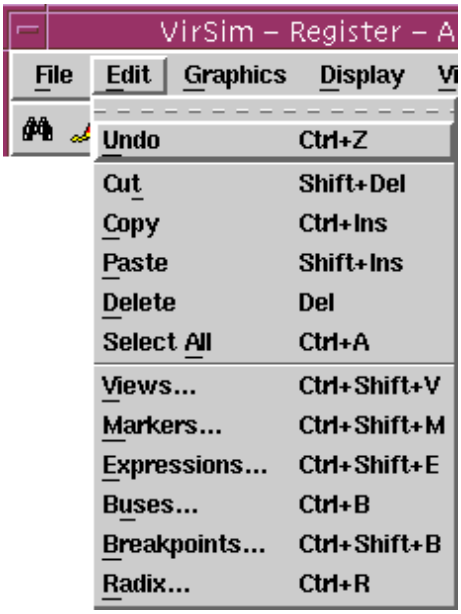
Markers - Opens the Markers Dialog. The dialog is used to define markers for specific simulation times.

Expressions - Opens the Expressions Dialog. The dialog is used to define and enable expressions for breakpoint groups.

Buses - Opens the Bus Builder Dialog.

Breakpoints - Opens the Breakpoint Groups Dialog. The dialog is used to create breakpoint groups for searching on enabled HDL expressions.

Radix - Opens the Radix Dialog. The dialog is used to create mnemonics, or user-defined radices.



Graphics Menu

The Graphics menu includes the following commands for positioning objects in the Register Window:

- Distribute Menu
- Align Menu

Align Menu

The following are commands for aligning selected objects in the Register Window. Select the objects you want to align. The Register Window aligns the selected objects with an anchor object. The last object you choose becomes the *anchor* object.

Left - Aligns objects with Left edge of anchor object.

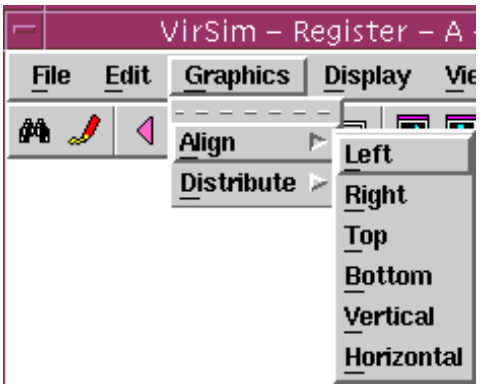
Right - Aligns objects with Right edge of anchor object.

Top - Aligns objects with Top edge of anchor object.

Bottom - Aligns objects with Bottom edge of anchor object.

Vertical - Aligns objects centered Vertically with anchor object.

Horizontal - Aligns objects centered Horizontally with anchor object.

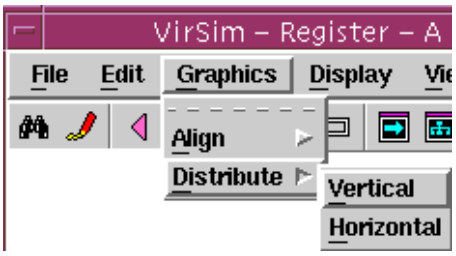


Distribute Menu

Using the Distribute commands, you can select and distribute objects equal distance between two points. The two points are determined to be the two extreme points of the selected objects.

Vertical - Distributes objects Vertically.

Horizontal - Distributes objects Horizontally.

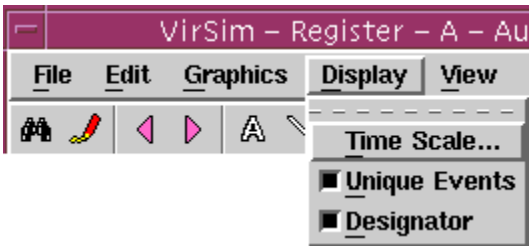


Display Menu

Time Scale - Opens the Time Scale Dialog for setting the display unit and display precision for VirSim display time.

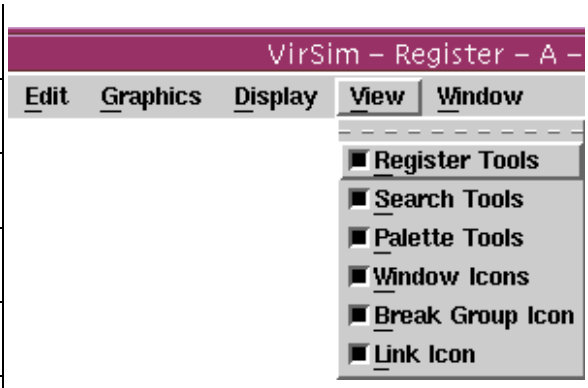
Unique Events - Toggles Unique Events *on* or *off*. When *on*, unique events display in the Register Window. When *off*, unique events do not display.

Designator - Toggles Designators *on* or *off*. When *on*, file designators appear ahead of the signal name. When *off*, file designators do not appear.



View Menu

View Menu settings are saved when you close VirSim.

Register Tools - Toggles display of the Register Window tool icons.	
Search Tools - Toggles display of the Search tool icons.	
Palette Tools - Toggles display of the Palette tool icons.	
Window Icons - Toggles display of the Window icons.	
Break Group Icon - Toggles display of the Breakpoint Group icon.	
Link Icon - Toggles display of the Link icon.	

8

Interactive Window

The Interactive Window is used to interactively control the simulator using a graphical interface. Using the Interactive Window you can send commands to and view the output of the simulator.

The Interactive Window can be synchronized with the other VirSim windows by using the link group SIM. When the Waveform Window, Source Window, Register Window, or Logic Browser are linked to the SIM link group, they can be used to monitor and control the simulation, as well as dynamically set interactive breakpoints.

The window's ability to operate on selected nets or scopes can significantly reduce the need for typing.

Interactive simulation results are available afterwards in a vpd file for post-processing debugging. For Verilog the default vpd file is vcdplus.vpd: for VHDL the default vpd file is inter.vpd. You can specify the vpd file name using +simargs+ on the command line.

Introducing the Interactive Window

The Interactive Window has the following areas.

- History Pane
- Command Prompt
- User-Defined Buttons
- Simulator Controls
- Window Status

History Pane

The History Pane displays the history of all commands sent to the simulator and a simulation log. By default, the scrolled history window displays the last 1000 lines of information. The value can be changed using `saveLines` as described in Common Settings on page 16-4.

Command Prompt

The command prompt is used to enter simulator commands during the course of the session. Click left in the Command region, enter the command, and press Return.

User-Defined Buttons

The user-defined buttons in the display area are used to send commands to the simulator. Yellow text buttons denote that a command applies to a selected object. White text buttons do not need a selected object. All user-defined buttons may be changed. See Controlling Simulation from Buttons on page 8-16 for more information.

Simulator Controls

The Simulator Control Area has the following controls:

- Step Control
- Scope Control
- Time Display

Step Control

Select Step Time, Step Signal, or Go To Time to run a simulator for a specified time, to a change on specified signal, or to a specified time.

Step Button Menu	Entry Field Contains
Step Time	Time Units
Step Signal	Signal name
Go To Time	Time Units

Scope Control

The Scope Control is used to specify and display the current scope. Entering a scope and clicking **OK** is equivalent to using a `$scope()` command.

Time Display

The Time Display shows the current simulation time and time units. The time units are read from the simulator. For Verilog, initially the units are the unit from the last timescale directive compiled. A `$timeformat()` can be used to override this value.

VirSim ignores precision and units string specified in the `$timeformat()` command.

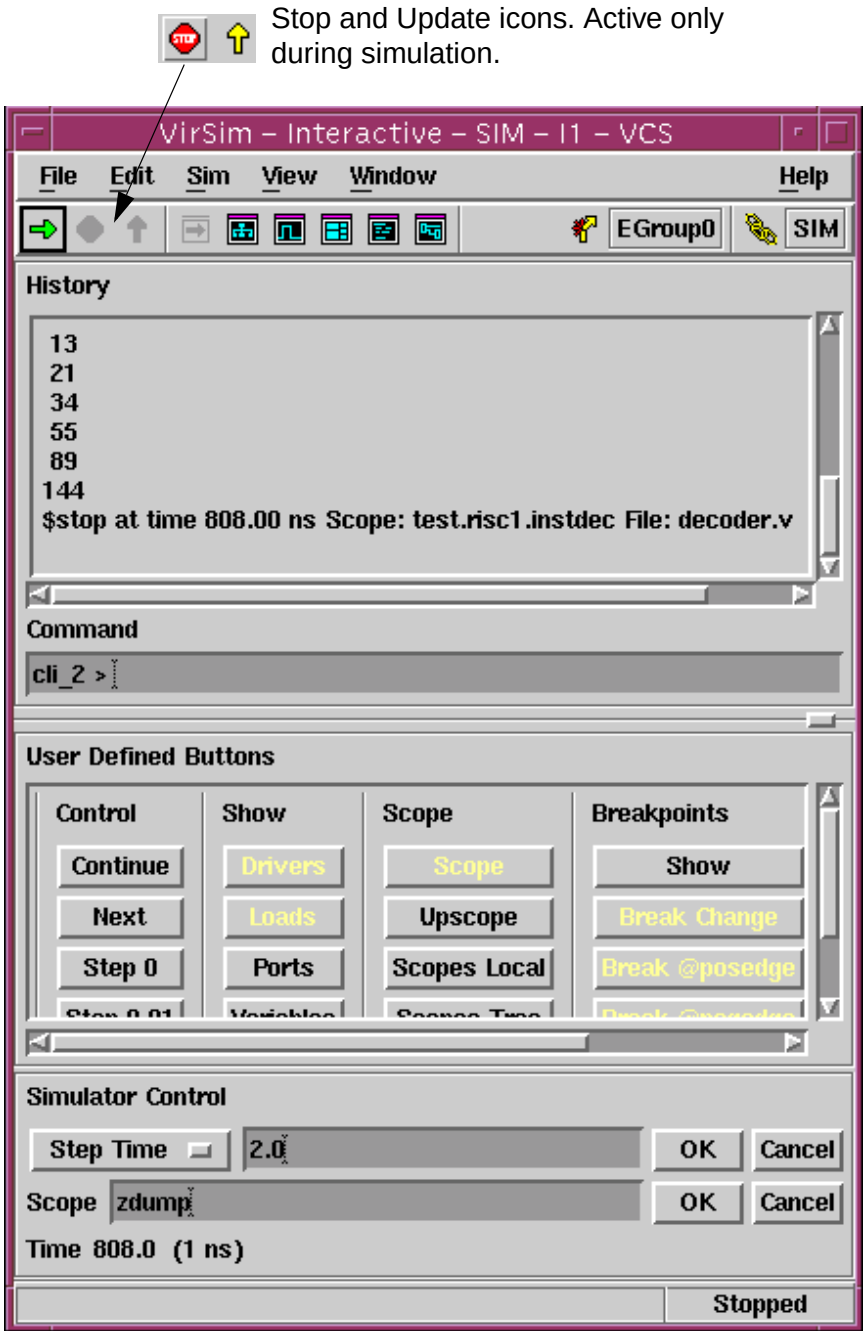
Note:

`$timeformat()` does not take effect until the statement is executed. A `$timeformat()` entered interactively takes effect immediately. A `$timeformat()` in an initial block, for example, only takes effect when it is executed. For example, `#30 $stop;` entered interactively may mean 30 ps before a `$timeformat()` command in an initial block is executed, but 30 ns after. This can cause unexpected, but correct, results during start-up.

Window Status

This status bar gives a brief description of the object pointed at.

Figure 8-1 Sample Interactive Window



The Interactive Window provides menu and toolbar controls, simulation log, simulator control, and a pane containing pre-defined and user-defined buttons.

Invoking an Interactive Session

The following information is covered in this section:

- Starting a Session Via the Simulator Invocation Dialog
- Starting a Session Via the VirSim Command Line

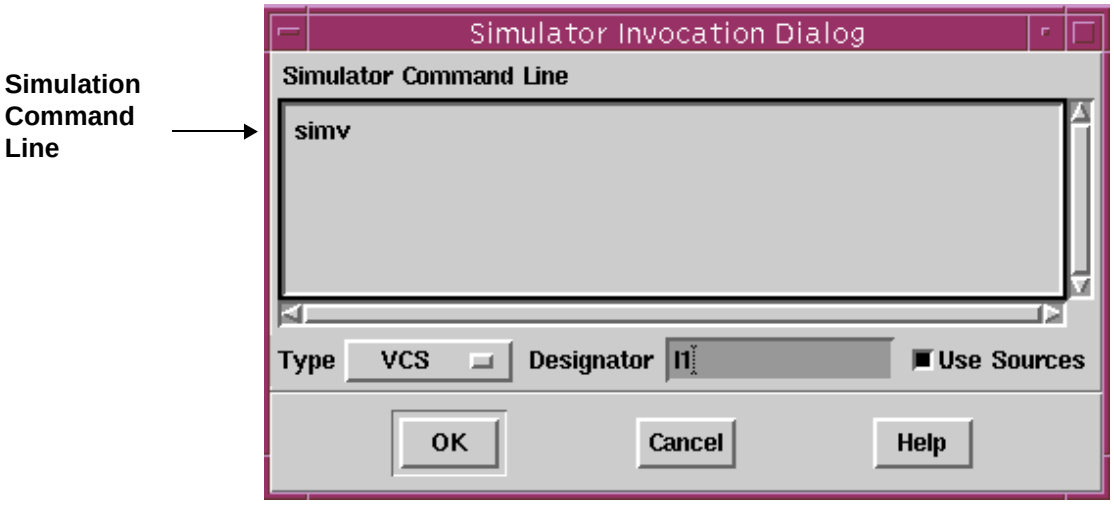
Starting a Session Via the Simulator Invocation Dialog

The Simulator Invocation Dialog (Figure 8-2, Simulator Invocation Dialog) is used to invoke the simulator in interactive mode and specify the simulation command line arguments. It opens when you select the Interactive button in the Main Menu.

Note:

You can invoke the Interactive Window from the Simulator Invocation Dialog or directly from the VirSim command line.

Figure 8-2 Simulator Invocation Dialog



The Simulation Command Line text area is used to invoke the simulator in an interactive session. Click left in the Command Line area, enter or modify the simulator invocation command, and click left on **OK**.

When enabled, Use Sources allows VirSim to use Verilog sources as supplied on the command line. Sources may be required by the Source Window, Logic Browser, or Event Origin command.

Note that VirSim only allows one set of sources on the command line. For example, you may open a VCD+ file for design XYZ then start an interactive session for design ABC. Since VirSim only allows one set of sources on the command line, only one of the designs should have the Use Sources button set to On.

Use Sources is enabled by default for VCD+ files. Use Sources is not sensitive for EPIC files because the VirSim EPIC interface does not allow sources. For VHDL VCD+ files the Use Sources button is inactive.

Starting a Session Via the VirSim Command Line

You can change the default simulator and invocation arguments on the VirSim command line by using the following options:

1. `+simtype+<simulator_type>`, where simulator_type is VCS, Scirocco, or VXL.
2. `+sim+<simulator_path>`, where simulator_path is a path to your simulator executable (for example, `myvxl`, `mysimv`, or `/usr/local/mysim`).
3. `+simargs+<simulator_option> / +simargs+"<simulator_options>"`, where *simulator_options* could be "`+define+DUMPVPD +vpdfile+myfile.vpd`".

For example:

```
virsim source.v +simtype+VCS +sim+/usr/local/my_simv
```

If the above command line options are used, the Interactive Window immediately opens. The above options can be combined with the `+cfgfile+<cfg_name>` option to also load a configuration file.

Depending on the platform, you also can change the defaults in the X Resources file on Unix or the registry on NT. See Common Settings on page 16-4 for more information.

Note:

Simulation invocation entries must be a full path to your simulator. There should not be environment variables or script names.

Controlling the Simulator

This section describes several ways to start and control the simulator. The simulator runs forward until it is stopped (using the Stop icon or Ctrl+C), finishes, or a breakpoint is reached. You can set breakpoints from the Interactive Window or any other window linked to the SIM link group.

To go backwards in a window, that window first must be unlinked from SIM.

The following information and procedures are covered in this section:

- Summary of Simulator Control Commands
- Breakpoint Durations
- Expression Breakpoints
- Control Simulation from the Waveform Window
- Control Simulation from the Source Window
- Control Simulation from the Logic Browser
- Control Simulation from the Register Window

Summary of Simulator Control Commands

Simulator control commands set the run time interval, run the simulator to the next change on a specified signal, or run to a specified simulation time. You can choose the appropriate commands from a menu in the Simulator Control pane. The following commands are available for most simulators.

Step Time	The Step Time command runs the simulator for a specified time interval and then stops. The specified time must be between 0 and the maximum simulation time.
Step Signal	The Step Signal command runs the simulator to the next value change of the specified signal and then stops.
Go To Time	The Go To Time command runs the simulator to the specified time and then stops. the specified time must be between 0 and maximum simulation time.

Breakpoint Durations

There are three different durations for breakpoints:

1. Transient breakpoint: These are automatically cleared the next time the simulator stops.
2. Temporary breakpoint: These are set until the breakpoint is satisfied or until manually cancelled.
3. Permanent breakpoint: These must be manually cancelled to remove the breakpoint.

Setting a line breakpoint or searching on an HDL expression are examples of setting a permanent breakpoint. The simulator controls described below set different types of breakpoints.

Expression Breakpoints

In the Waveform Window, use the Expression Dialog and Breakpoint Group Dialog to create the desired HDL expressions to search for breakpoint groups. For each desired expression, click the Breakpoints toggle button on to enable searching. After the expression breakpoints are set, continue the simulator. As soon as one of the HDL expressions is found, the simulator stops and displays the updated results in all windows linked to the SIM link group and any Waveform Window.

Note:

The VHDL simulator handles only true and false expressions, whereas VirSim and Verilog allow expressions based on signal value. If you are using VHDL or mixed VHDL/Verilog with a Scirocco simulator, be sure that expressions are written for a True/False result. For example, use (signal1 or signal2) = '1' rather than (signal1 or signal2).

For information on using breakpoint groups, see Locating Events Using Breakpoint Groups with Enabled Expressions. For information on creating expressions, see Chapter 15, Expressions.

When the simulator stops because of an expression breakpoint, a message displays in the History area of Interactive Window. For example:

```
Simulator stopped on expression search: expr_name
```

The name of the expression satisfied is displayed; in the above example, it is `expr_name`.

Control Simulation from the Waveform Window

There are two ways to control the simulator from the Waveform Window.

1. Search Next

   Causes the simulator to continue to the next breakpoint.

2. Go to a cursor or a marker

   The Markers icon in the toolbar pops up a list of cursors and markers. If the time of the marker or cursor is greater than the current simulation time, selecting the marker sets a breakpoint at the marker time and causes simulation to continue.

   Note:

   A convenient way to go to a time is to set C2 at the desired time then select C2 from the toolbar Marker menu. Keep in mind that the simulator will stop at any other previously set breakpoint.

3. Go to the Next Edge or Previous Edge of a selected signal or group of signals.

   Position the mouse in the Waveform Pane and right click to open the CSM. Select Next Edge or Previous Edge from the CSM. VirSim locates the next or previous edge that occurs in any of the selected signals.

Note:

Find Next/Previous Edge also stops on any expression breakpoint that is turned on in the currently active breakpoint group.

Control Simulation from the Source Window

There are four ways to control the simulator from the Source Window. Note that in the discussion below, all breakpoints refer to the currently active Instance Group. Only one Source Window may be linked to the simulator at any point in time, and only one Instance Group may be active.

1. Next Line (Transient breakpoint)

   The Next Line icon steps the simulator to the next line executed in any module in the current instance group.

2. Next Breakpoint (Permanent breakpoint)

   The Breakpoint icons run the simulator to the next breakpoint in the current instance group.

3. Time (Temporary breakpoint)

   Enter a time greater than the current simulator time in the Time field and select **OK**. This sets a breakpoint for an absolute time breakpoint and causes the simulator to continue.

4. Marker (Temporary breakpoint)

   The Markers icon in the toolbar pops up a list of markers. If the time of the marker or cursor is greater than the current simulation time, selecting the marker sets a breakpoint at an absolute time and causes simulation to continue.

Note:

When controlling the simulator from the Source Window, the simulator runs until it encounters any breakpoint. Therefore, the simulator may stop before the desired point for one of two reasons: (1) a breakpoint was entered directly in the simulator command line, or (2) an expression breakpoint was active in the breakpoint group and evaluated as TRUE.

Line Breakpoints

After a scope is loaded in the Source Window, green dots are displayed in the left side of the Source Window to represent executed lines. Each executed line can have a breakpoint. To set a line breakpoint, click left on the green dot. The dot changes to red when the breakpoint is set. The line breakpoint is permanent until removed.

If the Source Window is linked to the SIM link group, the line breakpoints set simulator breakpoints. After all breakpoints are set, you can continue the simulator. As soon as the simulator executes a line with a breakpoint, it stops and displays the updated results in all windows.

When the simulator stops because of a line breakpoint, a message displays in the History area of the Interactive Window. For example:

```
Simulator stopped at line breakpoint: Line 12, Module top
```

If the Source Window is not linked to the SIM link group, the line breakpoints do not stop the simulator.

Control Simulation from the Logic Browser

(Verilog only) There are three ways to control the simulator from the Logic Browser.

1. **Next Selected Net** (Transient breakpoint) - Causes the simulator to continue until the selected net changes.
2. **Next Any Change** (Transient breakpoint) - Causes the simulator to continue until there is a change on any of the displayed signals.
3. **Time** (Temporary breakpoint) - Sets a breakpoint at an absolute time and causes the simulator to continue. Enter a time greater than the current simulator time in the Time field and press **OK**.

Control Simulation from the Register Window

There are three ways to control the simulator from the Register Window.

1. **Markers** (Temporary breakpoint) - The Markers icon in the toolbar pops up a list of bookmarked times. If the time of the marker or cursor is greater than the current simulation time, selecting the marker sets a breakpoint at an absolute time and causes simulation to continue to the marker time.
2. **Next Any Change** (Transient breakpoint) - The simulator runs until a signal in the Register Window changes state value.
3. **Time** (Temporary breakpoint) - Sets a breakpoint at an absolute time and causes the simulator to continue. Enter a time greater than the current simulator time in the Time field and press OK

Controlling Simulation from Buttons

The scroll area titled User Defined Buttons consists of buttons for which you can configure behavior, layout, and color.

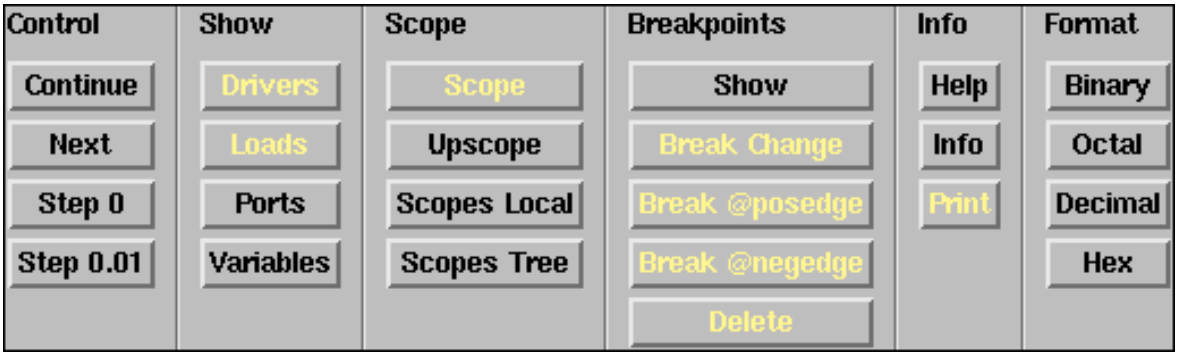
The following information is covered in this section:

- Default Buttons Files
- Creating User-Defined Buttons
- Defining a Button
- Selection Substitution

Default Buttons Files

The buttons are defined in ASCII button files named `iwbuttons.<sim>` in `$VIRSIMHOME/appfiles`, where <*sim*> is a 3 character extension to identify the simulator. You can modify this file to add buttons for frequently used commands. The Button Area is divided into sections with each section identified by a section title (first row) and any number of buttons. Figure 8-3, Pre-Defined Button Layout for a Verilog Simulator, shows the default for a Verilog simulator.

Figure 8-3 Pre-Defined Button Layout for a Verilog Simulator



The buttons file is searched for in the following locations (in order).

- The current directory
- The users HOME directory
- The installation directory

The button definition file is used to define button sections. In Figure 8-3, Pre-Defined Button Layout for a Verilog Simulator, there are five button sections. After a button section is defined you may define buttons for each section. These buttons may explicitly specify the command to be executed or may use a selection substitution. See Selection Substitution later in this chapter.

Note:

On Windows NT, the button files must be located in the APPFILES subdirectory under VIRSIMHOME.

Creating User-Defined Buttons

The following is an example of a button section in the button file. On Windows NT, colors such as White Smoke are identified by hexadecimal numbers.

The title displayed in the User-Defined Button area is Control Buttons. It has two columns of buttons and the title text uses the color White Smoke. The label, numcolumns, and foreground are all options. If unspecified, the following defaults are used:

```
label = same as section name
numcolumns = 1
foreground = default foreground in X Resource file
```

The individual options are separated by commas, and the entire section definition is terminated with a semicolon.

Note:

In X Resource files, if the color specified has embedded spaces, it is necessary to enclose the color in double quotes.

Defining a Button

The following is an example of a button definition:

```
# Uses red for foreground and black for background
define button Control
     label "Save",
     command "$save("saved.exe")",
     foreground "Red",
     background "Black";
```

The above definition creates a user defined button in the Control section. The button label is **Save**. When the button is pressed, the command `$save("saved.exe")` is sent to the simulator. The button foreground is red and the background is black.

The label and command options are required; the foreground and background are optional. If alternative settings are not specified, the options use the default window settings.

The individual options are separated by commas, and the entire button definition is terminated with a semicolon.

Selection Substitution

A button definition may contain a command that includes a `$*`. When you execute the button, this special string is replaced with the currently selected text. For example, the button definition file may specify the command `$display($*)` to select a signal and click the button. If you select `top.m0.a` then click the button, the command `$display(top.m0.a)` is sent to the simulator.

Note:

To help distinguish between normal buttons and the buttons that have selection substitution, it is recommended that all special buttons have a special foreground color. The default button file shipped with this product sets the foreground color to khaki for emphasis.

Toolbar and Menu Reference

Toolbar

You can toggle display of toolbar icons from the View menu.

Go Icon	The Go icon causes the simulator to continue. It is the same as selecting Continue from the Sim pull down menu.
Stop Icon	The Stop icon stops the simulator. It is the same as selecting Stop from the Sim menu or entering Ctrl+C.
Update Icon	When the simulator is running, the Update icon loads data from the RAM buffer. Normally, the data in RAM is only loaded when the simulator stops.
New Window Icons	These icons allow you to open new VirSim windows from the Interactive Window toolbar.
Breakpoint Group Icon	The Breakpoint Group icon opens a pop-up list of available breakpoint groups. Breakpoint groups consist of groups of user-defined HDL expressions that may be enabled for searching. You can select available breakpoint groups from the menu. Breakpoint groups are created in the Breakpoint Groups Dialog. The expressions are created in the Expressions Dialog. For information on creating breakpoint groups, see Locating Events Using Breakpoint Groups with Enabled Expressions on page 4-24.
Link Icon	The Link icon is similar to the Link icon in the Source, Waveform, Logic, and Register windows. These windows can link to the SIM link. Their time then is synchronized with the simulator, and they also can control the simulator. The Interactive Window is always linked to the SIM link and cannot be changed.

Menu Bar

Note:

Any menu with a dotted line at the top can be torn off and placed on the display for easy access. Click middle above the dotted line, and then drag the menu to a convenient position.

File Commands

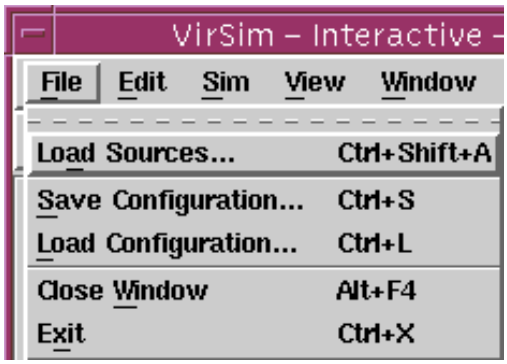
Load Sources - Displays the Load Sources dialog to compile Verilog source.

Save Configuration - Displays the Configuration dialog to save the current configuration.

Load Configuration - Displays the Configuration dialog to load a configuration file.

Close Window - Closes the Interactive Window.

Exit - Exits VirSim.



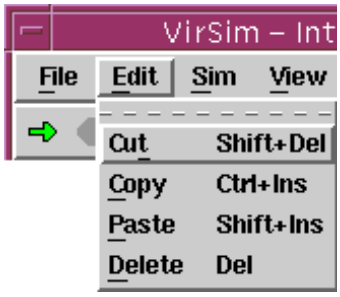
Edit Commands

Cut - Cuts the selection and moves it to the clipboard.

Copy - Copies the selection to the clipboard.

Paste - Pastes the clipboard contents to a designated area.

Delete - Deletes the selection.



Sim Commands

Continue - Continues to run the simulator.

Stop - Stops the simulator.

Finish - Finishes the simulation.

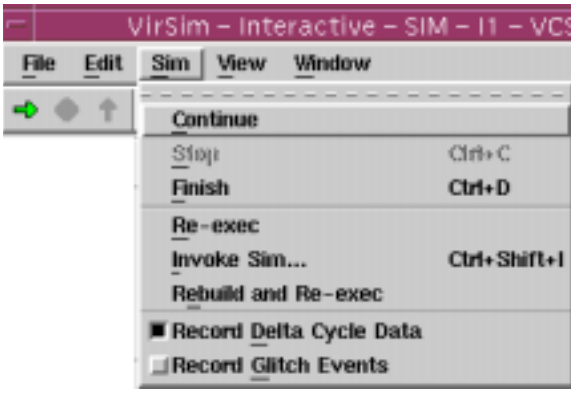
Re-exec - Reruns the simulation with same arguments as the previous run. If a simulation is currently running, it automatically terminates.

Invoke Sim... - Opens the Simulator Invocation Dialog to start a new simulation or edit simulation options.

Rebuild and Re-exec - Rebuilds and reruns the simulator without exiting from VirSim. (For use with compiled code simulators.)

Record Delta Cycle Data - Toggles recording of delta cycle information while running in interactive mode.

Record Glitch Events (Verilog only.) - Toggles recording of glitch information while running in interactive mode.



View Menu

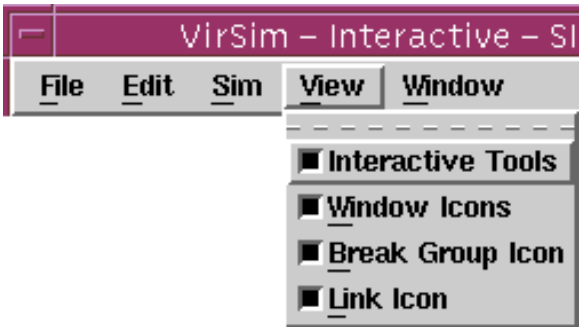
View Menu settings are saved when you close VirSim.

Interactive Tools - Toggles display of the Interactive Window tool icons.

Window Icons - Toggles display of the Window icons.

Break Group Icon - Toggles display of the Breakpoint Group icon.

Link Icon - Toggles display of the Link icon.



9

Project Window

The Project Window is used to analyze, elaborate, and simulate VHDL or mixed VHDL and Verilog projects.

Introducing the Project Window

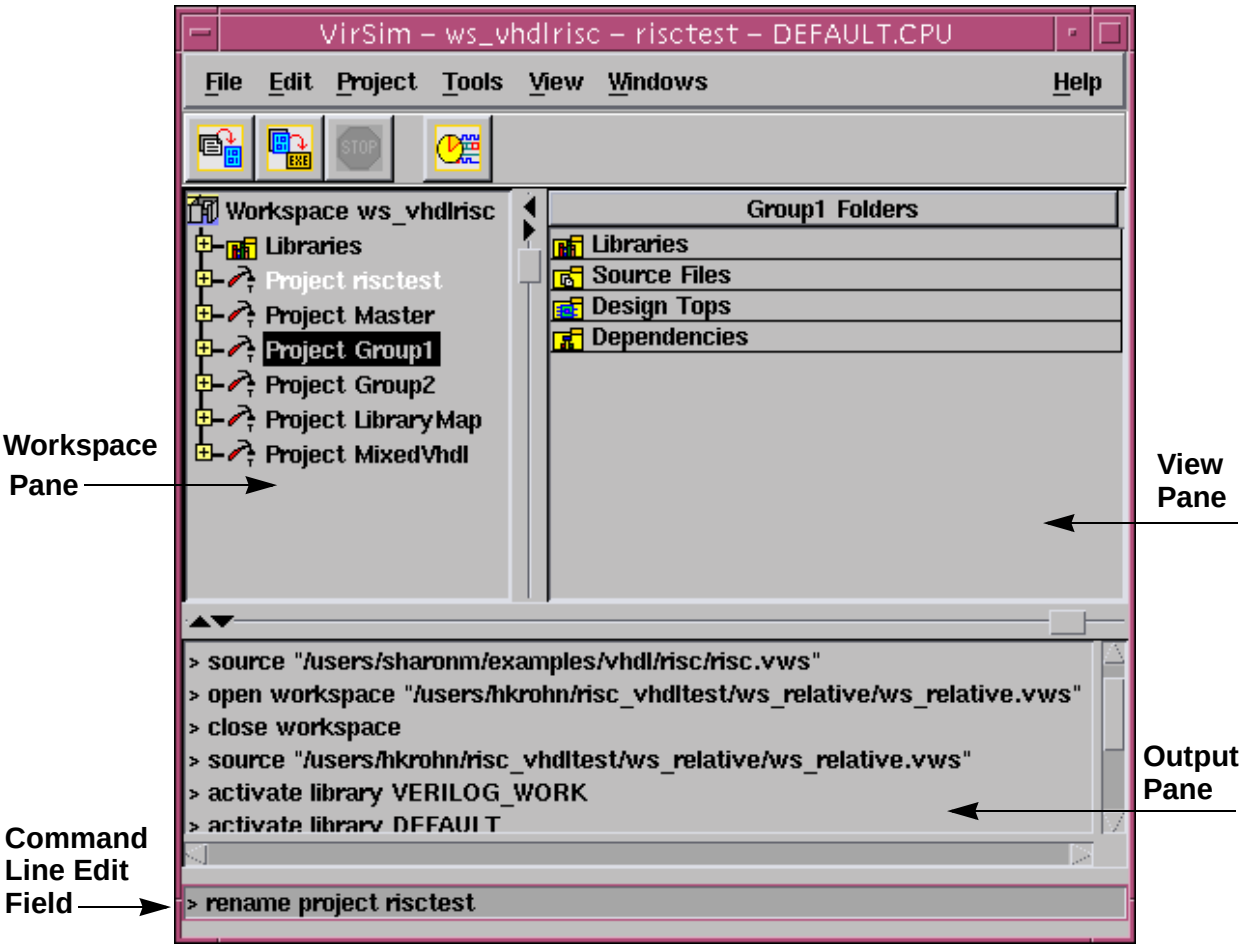
The Project Window has three panes as shown in Figure 9-1, Sample Project Window.

- Workspace Pane - Displays the content of a Workspace.
- View Pane - Displays the contents of Folders and Libraries.
- Output Pane - Displays analyze and elaboration output including error messages.

In addition to the three panes, the Project window includes the following areas:

- Title bar - Displays the name of the open workspace, the name of the active project, and the name of the active design top.
- Toolbar - Contains analyze, elaborate, stop and simulate icons.
- Command Line - Allows you to execute commands and scripts without using the GUI.

Figure 9-1 Sample Project Window



Defining Projects and Workspaces

The Project Window displays a workspace and a number of projects. Only one workspace can be open in the Project window.

What is a Workspace?

A workspace contains one or more projects. A workspace has settings that can be shared by all projects in the workspace. You can have one workspace or multiple workspaces. A workspace can contain all of your projects or there can be a workspace for each project.

A workspace containing multiple projects can have only one active project. The active project is the project being analyzed, elaborated, or simulated. Finally, a workspace can have one active design top. The active design top belongs to the active project and is the design unit currently being elaborated or simulated.

A directory is created for each new workspace if the directory does not exist. Child directories and files used by the Project Window are created in the workspace directory. A new directory beneath the workspace directory is created for each project. The project directory contains a file for recreating the project, an analyze history file and a library directory named WORK. The library using the directory WORK is named DEFAULT.

What is a Project?

A project contains VHDL and Verilog source file paths, libraries that receive analyzer output, references to other projects that might need to be analyzed when the project is analyzed, and names of design units that are used as design tops for elaboration and simulation.

VHDL and Verilog source files are added to a project. The project is analyzed and analyze errors are corrected. The project is then elaborated and simulated. To troubleshoot the design, open individual Virsim Windows.

Multiple projects can be used to partition a design into manageable units and to build multiple libraries for a design.

What is a Design Top?

A design top is an entity, configuration or package declaration. The active design top is the design unit that will be passed to the Analyze or Simulate command.

To add a design top to a project first analyze the project, and then view the contents of the library containing analyzer output (right-click on the library name). The view displays entities, package declarations, and configurations. Use the View popup menu to add a design top to the project.

Workspace Pane

The workspace pane displays the content of a workspace. The pane allows you to expand and collapse the Workspace folder, Project folders, and Library folders. The content of the selected folder is displayed in the View pane.

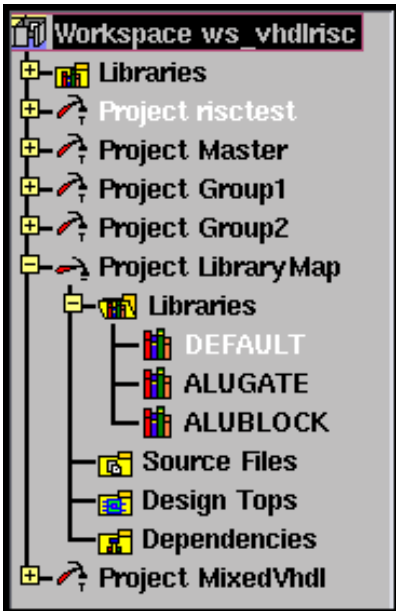
A workspace contains projects and settings common to all the projects. For example, all projects in a workspace can share certain analyzer settings.

Projects contain source files, design tops, libraries and names of other projects (dependencies) that might need to be analyzed if changes occur. Projects also contain analyze, elaborate, and simulator command line information.

Each project contains a DEFAULT library that is created when the project is created. Other libraries may be added to a project. There is always an active library in a project. Analyzer output goes to the active library, unless the Options Dialog has been used to specify the library that will receive analyzer output.

The active project and active library in a project are colored gold.

Figure 9-2 Workspace Pane



When a new project is created, the new project is displayed in a text edit field with a default name (see Figure 9-3, Renaming a Project). To accept the name, click the mouse outside the text edit field.

Change the new project name by clicking the mouse in the text edit field. Delete the old name and type in a new name, then press the enter key. The folder name “Project” cannot be removed. To cancel editing and accept the original name press the ESC key.

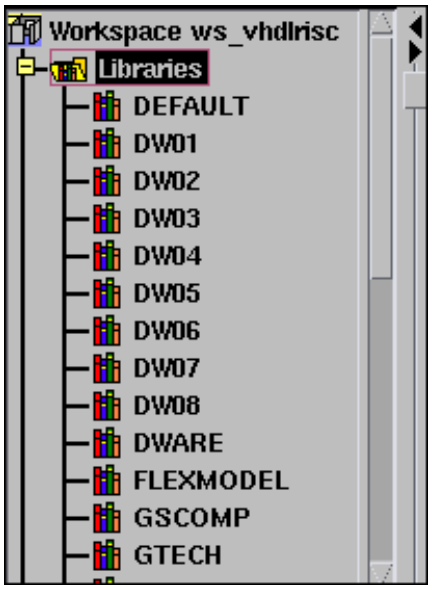
Figure 9-3 Renaming a Project

An expanded view of the workspace Libraries folder is shown in Figure 9-4, Workspace Library.

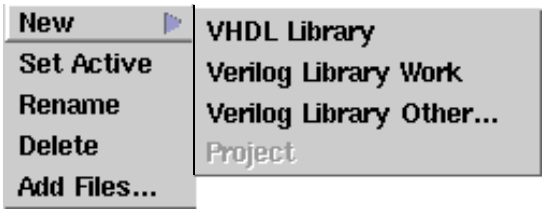
The Workspace library folder contains Libraries known to Scirocco and libraries defined by the user. When a workspace is opened, VirSim executes show_setup to obtain libraries known to Scirocco. (show_setup is part of the Scirocco installation.) Libraries defined in the synopsys_sim.setup files found in the Scirocco installation and in your home directory are found by show_setup.

A Scirocco library logical name can be changed using rename. The new library mapping appears in the synopsys_sim.setup file created by the project browser when analyzing or simulating a project (see Library Folder View.) Deleting the renamed library restores the original Scirocco library.

Figure 9-4 Workspace Library



Workspace Pane Context Sensitive Menu



New VHDL Library
Creates new workspace or project VHDL library. Shows default library name in text edit field. The VHDL library has a name and directory path.

New Verilog Library Work
Creates new project Verilog library Work. The library is named VERILOG_WORK. Verilog libraries are placed in the VERILOG_WORK library during Verilog compilation.

New Verilog Library Other...
Opens the New Verilog Dialog that is used to create Verilog libraries for the project. The Verilog libraries are entered in the synopsys_sim.setup file.

New Project
Creates a new project. Sets project as active project and displays default project name in text edit field.

Set Active
Sets project as active project. Double clicking a project also sets a project as the active project.

Rename
Renames a project or library. Project or library name is displayed in text edit field and you can modify the name.

Delete
Deletes a project or library.

Add Files...
Opens Add File dialog so you can add files to a project.

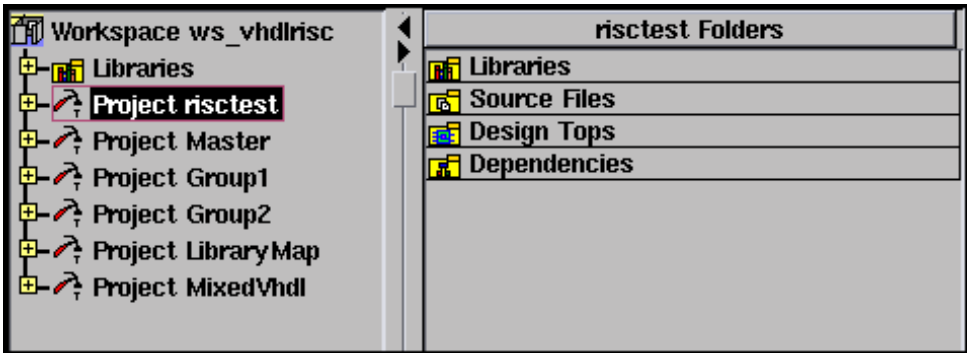
View Pane

The View pane displays contents of folders and libraries. There are six folders that can be displayed in the View Pane: Workspace, Project, Library, Source Files, Design Tops, and Dependencies. To view the folders content, double click a folder in the View Pane.

Workspace Folder View

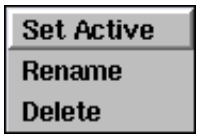
The Workspace Folder View displays the workspace library folder and project folders in the workspace. The active project is displayed with a unique color.

Figure 9-5 Workspace Folder View



Workspace Folder View Context Sensitive Menu

This menu is displayed when pressing the right mouse button.



Set Active
Sets a project as the active project.

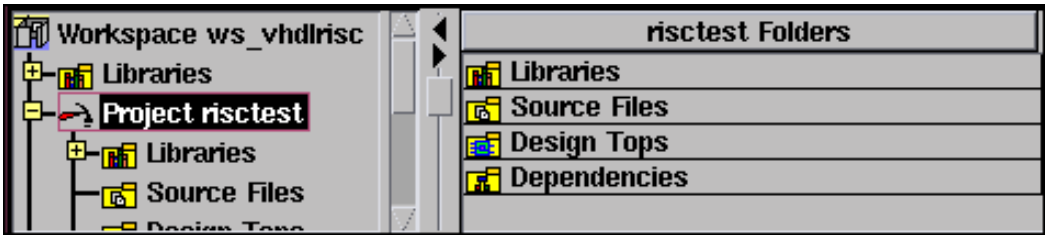
Rename
Renames a project. The project name is displayed in a text edit field in Workspace View.

Delete
Deletes a project. Selecting the project and pressing the Delete key also deletes project.

Project Folder View

The four project folders are displayed in Figure 9-6, Project Folder View. To view its content, double click a folder.

Figure 9-6 Project Folder View



Project Folder View Context Sensitive Menu

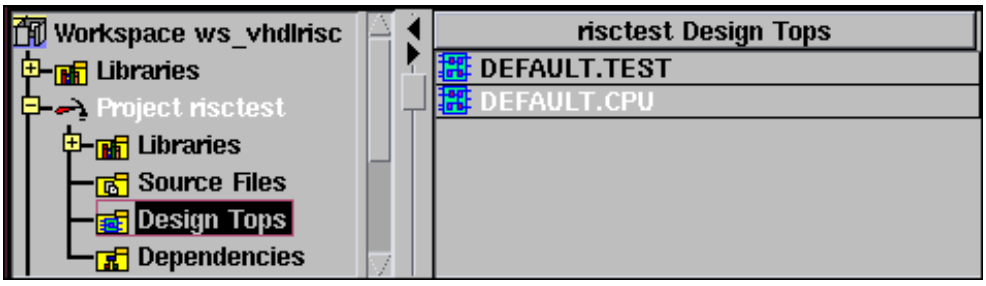
Set Active
Sets a project as the active project.

Add Files...
Opens Add File dialog so you can add files to active project.

Design Top Folder View

This view displays design tops contained in the projects design top folder. The active design top is displayed with a unique color.

Figure 9-7 Design Top Folder View



Design Top Folder View Context Sensitive Menu

Delete
Deletes design top from the project. A design top can also be deleted by pressing the Delete key after the design top is selected.

Set Active
Sets design top as the active top.

Source File Folder View

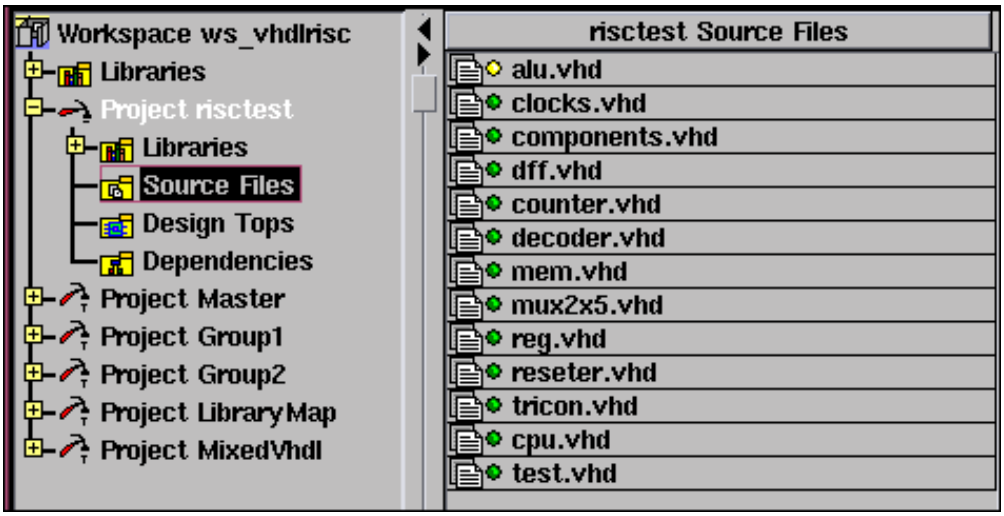
Display source files in the right splitter pane by double clicking the source files folder. Red, green, and yellow icons are displayed near the source file name.

- Yellow icon indicates that a source file needs to be analyzed.
- Green icon indicates that the source file was analyzed correctly.
- Red icon indicates an analyze error.

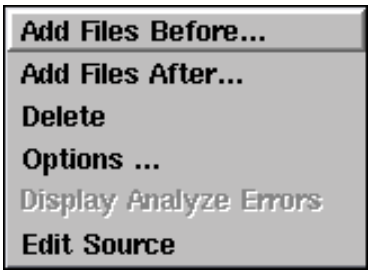
Source files are analyzed in the order displayed. Use the Source File View to reorder files. Select a file and use Ctrl-UpArrow or Ctrl-DownArrow to move the file.

The project contains source file paths, not a file copy. So, adding and deleting project source files does not remove the file from the file system.

Figure 9-8 Source File Folder View



Source File Folder View Context Sensitive Menu



Add Files Before...
Opens Add File dialog so you can add files to a project before file pointed to by mouse.

Add Files After...
Opens Add File dialog so you can add files to a project after file pointed to by mouse.

Delete
Removes file from project. A source file can also be removed by pressing the Delete key after the source file is selected.

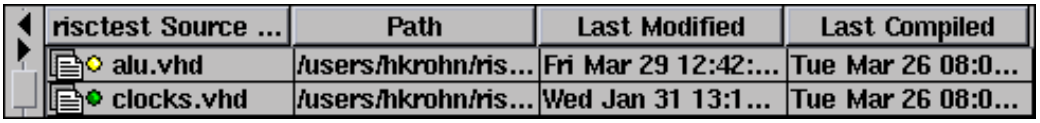
Options...
Opens Options Dialog which allows you to set analyze options for VHDL files or compile options for Verilog files.

Display Analyze Errors
Displays analyze errors in the output pane.

Edit Source
Displays the source file in an editing window.

Full Detail View

The full detail view shows the file name, Path, Last Modified Date and Last Compiled Date.

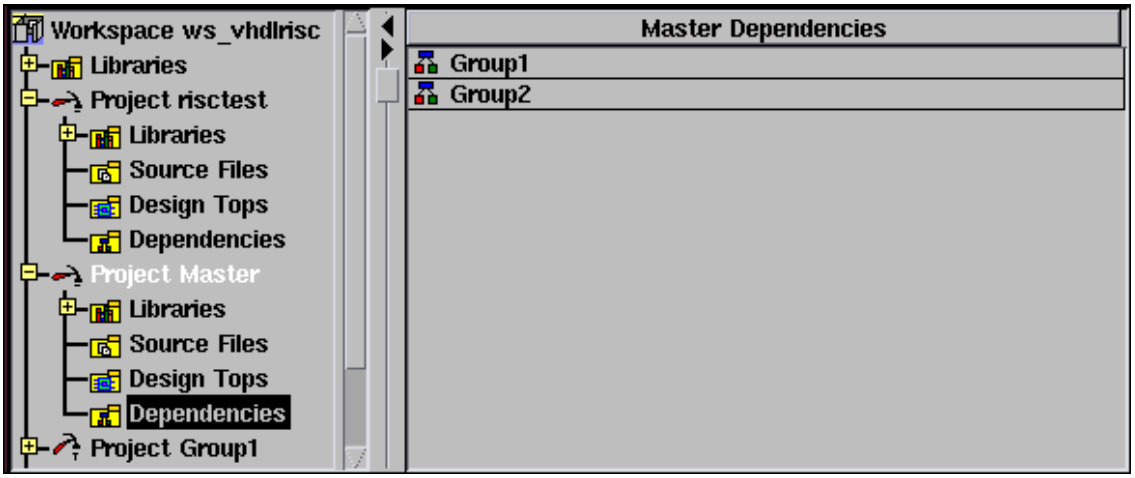


Dependency Folder View

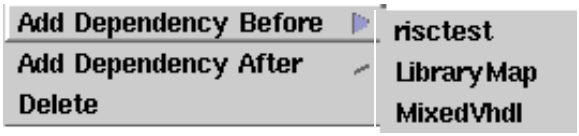
Dependent projects are analyzed in the order displayed and before source files in the project. Use the Dependency View to reorder files. Select a project name and use Ctrl-UpArrow or Ctrl-DownArrow to move the dependent project.

When an active project is analyzed, projects in the dependency list are analyzed if they are out of date. A project is out of date when its library is cleaned, a source file is modified, dependent projects are out of date, or analyze errors are detected.

Figure 9-9 Dependency Folder View



Dependency Folder View Context Sensitive Menu



Add Dependency
Adds the selected project from the submenu to the end of dependency list.

Add Dependency Before
Adds the selected project from the submenu before the selected project.

Add Dependency After
Adds the selected project from the submenu after the selected project.

Delete
Removes a dependent project. You can also use Delete key to remove a dependent project.

Library Folder View

The library folder view displays libraries in the workspace library folder or a project library folder.

You can use the Library Folder view to add libraries or rename libraries. Use the rename command to change individual library fields.

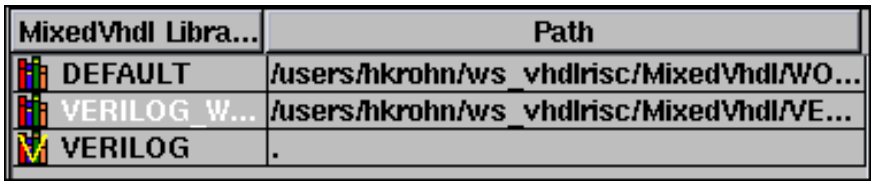
When a project is analyzed, elaborated, or simulated, the Project Window automatically creates a synopsys_sim.setup file. The file is created in the active project directory when a project is analyzed and in the simulation directory (Using the Options Dialog on page 9-39) when a project is elaborated or simulated. VirSim changes the directory setting to the project directory when analyzing or the simulation directory when elaborating or simulating. Scirocco uses the new setting from the synopsys_sim.setup file in the project directory.

Libraries in a projects Library Folder are added to the synopsys_sim.setup file created by the Project Window in the order that they appear in the Library Folder. Libraries in the workspace library folder that were created for the open workspace, or workspace libraries that are modified, are written to the synopsys_sim.setup file after project libraries.

You can create three types of libraries: VHDL libraries, Verilog libraries, or a special Verilog library named VERILOG_WORK. VHDL libraries and the VERILOG_WORK library are associated with a directory path with the form ’name:path’ in the synopsys_sim.setup file. Verilog libraries are associated with a path to a Verilog library file or Verilog library directory with the form ’name < path’ in the synopsys_sim.setup file. If the Library is a library directory, after the path name enter a colon (:) followed by one or more filename extensions to specify the extensions of the files in the library that you want searched (to resolve module instances). For example:

```
VLOGLIB1 < /net/cores/vloglib1:.v+.vp
```

Figure 9-10 Library Folder View



Library Folder View Context Sensitive Menu

New VHDL Library
Creates new workspace or project VHDL library. Shows default library name in text edit field. The VHDL library has a name and directory path.

New Verilog Library Work
Creates new project Verilog library Work. The library is named VERILOG_WORK. Verilog libraries are placed in the VERILOG_WORK library during Verilog compilation.

New Verilog Library Other...
Opens the New Verilog Dialog that is used to create Verilog libraries for the project. The Verilog libraries are entered in the synopsys_sim.setup file.

Delete
Deletes a library from a project or workspace.

Rename
Renames the library logical name or path.

When a library path is renamed, a button is displayed next to the text edit field containing the path. The button can be used to open the “Select a Directory” Dialog. Pressing the Ok button in the Dialog transfers the selected directory to the text edit field.

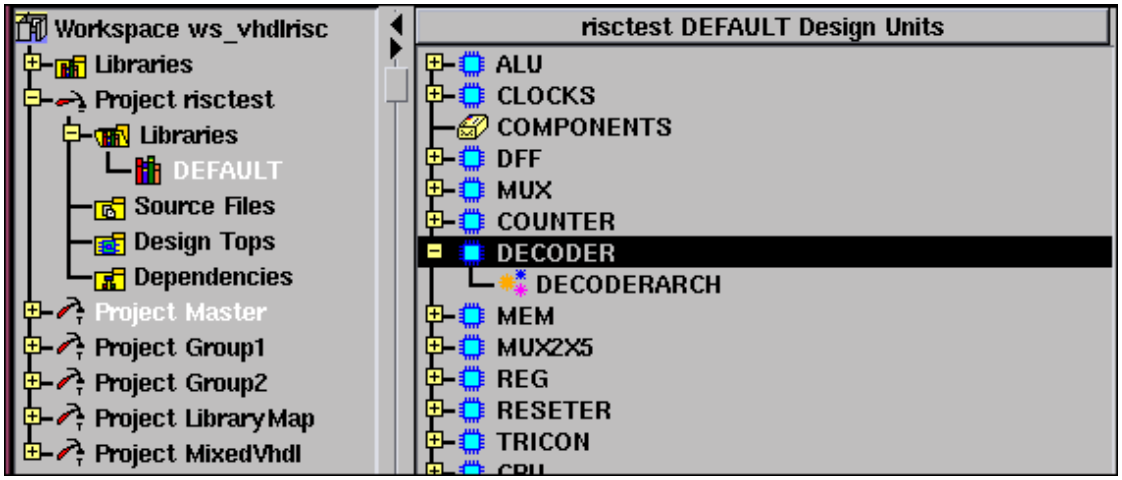
Set Active
Sets the library as the active library. The active library receives analyzer output when a library has not been specified using the Options Dialog.

Clean
Removes all files from the library.

Library View

The Library View displays the primary design units (entity, configuration, and package declarations) and secondary design units (architecture and package body’s). Secondary design units are shown by expanding the primary design unit.

Figure 9-11 Library View



Library View Context Sensitive Menu

Add Top

Adds a design unit to the active project's Design Tops folder. Sets this design unit as active design top.

Edit Source

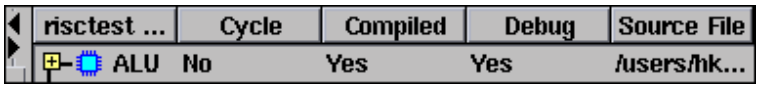
Displays source file containing design unit in text editor.

Clean

Removes all files from the library.

View Full Detail

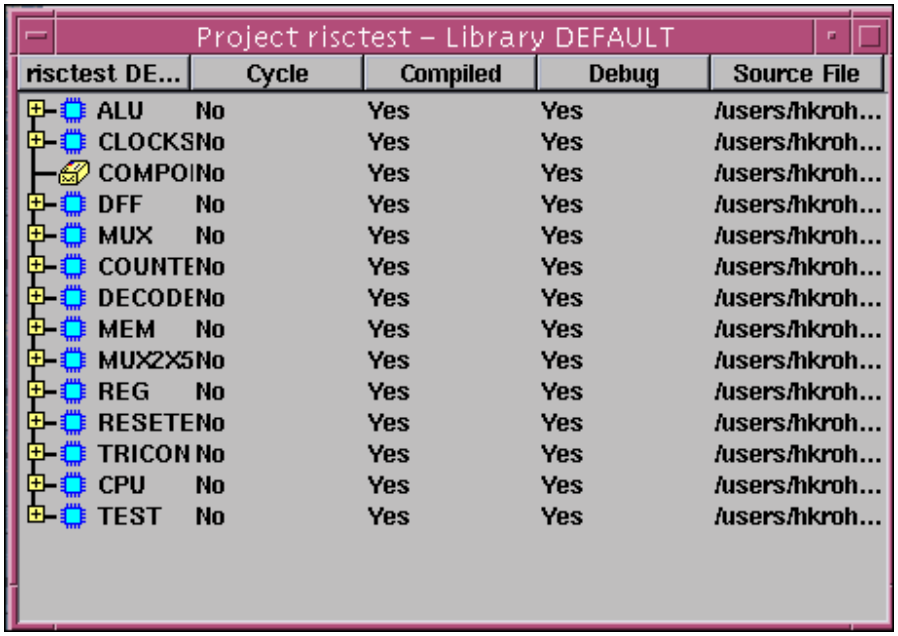
Displays the design unit, Cycle, Compiled, Debug flags, and the Source File.



Tear Off Windows

Any of the View Pane views can be displayed in a separate window. Use the View-Tear Off command from the Project Browser Main Menu to create a separate window containing the current View Pane View. Multiple Tear Off windows can be created. Context Sensitive menus are supported in Tear-Off windows.

Figure 9-12 Tear Off Window

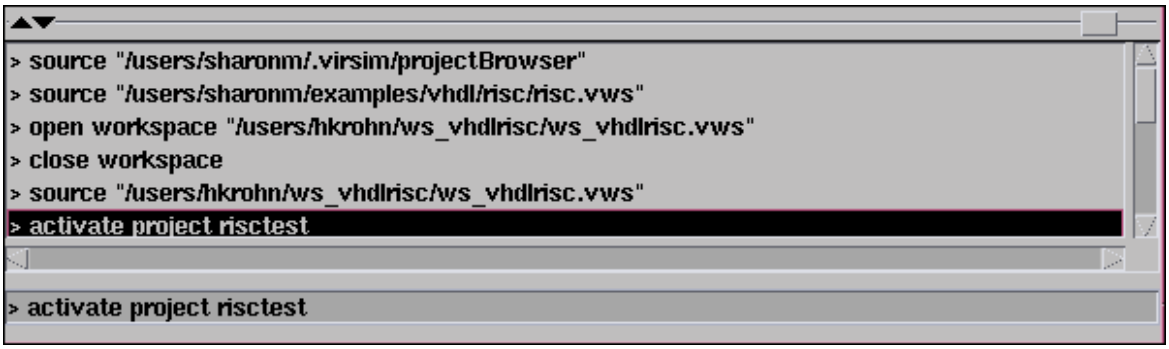

Project risctest – Library DEFAULT
risctest DE...
Cycle
Compiled
Debug
Source File
ALU No Yes Yes /users/hkroh...
CLOCKS No Yes Yes /users/hkroh...
COMPO No Yes Yes /users/hkroh...
DFF No Yes Yes /users/hkroh...
MUX No Yes Yes /users/hkroh...
COUNTE No Yes Yes /users/hkroh...
DECODE No Yes Yes /users/hkroh...
MEM No Yes Yes /users/hkroh...
MUX2X5 No Yes Yes /users/hkroh...
REG No Yes Yes /users/hkroh...
RESETE No Yes Yes /users/hkroh...
TRICON No Yes Yes /users/hkroh...
CPU No Yes Yes /users/hkroh...
TEST No Yes Yes /users/hkroh...

Output Pane

The Output pane displays a history of executed commands, analysis and elaboration output, and error messages.

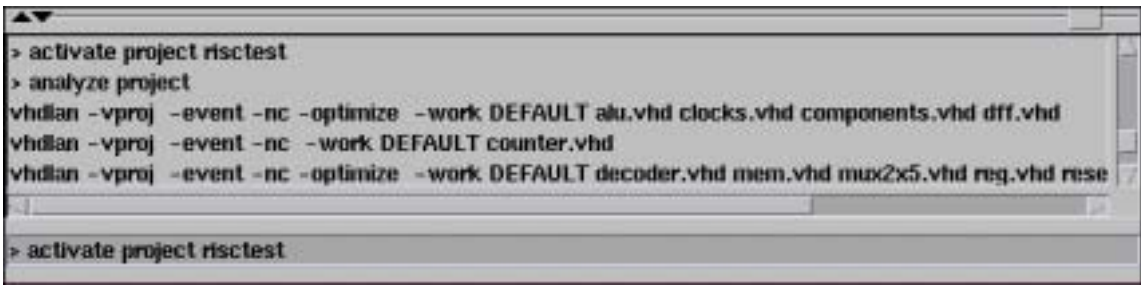
Commands can be transferred from the Output Pane to the Command Line Edit Field by double clicking the command in the Output Pane. In Figure 9-13, Output Pane With Commands, “activate project Group1” command is transferred to the command line by double clicking. The command can be edited further or executed by pressing the carriage return.

Figure 9-13 Output Pane With Commands



You can use the Output pane to read analyzer errors and to edit source files for correcting analyzer errors. Figure 9-14, Output Pane With Analyzer Output shows analyzer output and an error message. With the mouse over the error message, bring up the Context Sensitive Menu to edit the source or explain the error.

Figure 9-14 Output Pane With Analyzer Output


> activate project risctest
> analyze project
vhdlan -vproj -event -nc -optimize -work DEFAULT alu.vhd clocks.vhd components.vhd dff.vhd
vhdlan -vproj -event -nc -work DEFAULT counter.vhd
vhdlan -vproj -event -nc -optimize -work DEFAULT decoder.vhd mem.vhd mux2x5.vhd reg.vhd rese
> activate project risctest

Output Pane Context Sensitive Menu

Edit Source
Opens a source file in the text edit window.

Explain Error
Opens window with text explaining error.

Toolbar and Menu Reference

This section includes the following information:

- Toolbar
- Menu Bar

Toolbar

The Project Window toolbar contains the following icons.

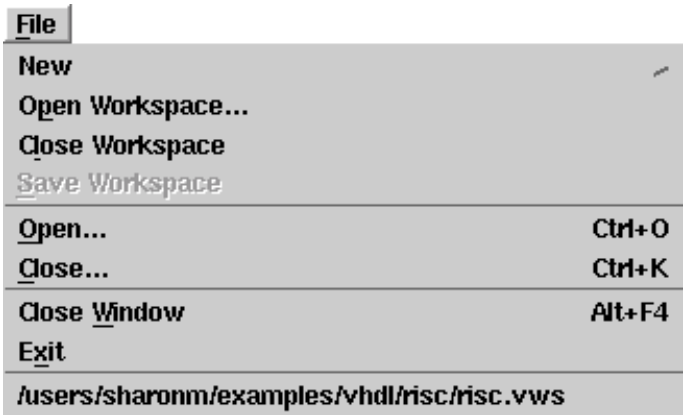
Analyze Icon	The Analyze icon analyzes the active project and all dependent projects. The message “Project is up to date” appears in the Output Pane when the project and all source files are previously analyzed and they do not need to be reanalyzed. Use Tools-Analyze All to unconditionally analyze a project. The project is always analyzed, even when it’s up to date.
Elaborate Icon	The Elaborate icon elaborates the active design top.

Stop Icon	The Stop icon stops the analyzer or elaborator. This icon is enabled when the analyzer or elaborator is running.
Simulate Icon	The Simulate icon opens the Virsim Interactive window, starts Scirocco, and simulates the active design top. Simulate options for the active design top are passed to the simulator. You can start Scirocco from the Interactive Window if it is open. However, using the Simulate Icon transfers the latest Simulate options to the simulator.

Menu Bar

File Menu

New - Opens a submenu for creating new workspaces, projects, and libraries. The description of this submenu immediately follows this description of the File menu.



Open Workspace… - Displays the Open File dialog. The dialog only displays *.vws files. The selected workspace file is opened in the Project window.

Close Workspace - Closes the open workspace.

Save Workspace - Saves the open workspace. Saving the workspace will create a workspaceName.wvs file in the workspace directory and a projectName.vpj file in each project directory. The .wvs and .vpj files contain commands that restore the workspace.

Open… - Displays the Open dialog to open an EPIC, VCD or VCD+ history file.

Close… - Displays the Close File dialog to close an open history file.

Close Window - Closes the Project Window.

Exit - Exits VirSim.

Previously Opened Workspaces - Displays a list of previously opened workspaces in the File Command Menu. Selecting one of the workspaces opens it.

New Workspace - Displays the New Workspace dialog. Use this dialog to create a new workspace.
New Project - Creates a new project. Sets the project as the active project and displays the default project name in a text edit field in the Workspace pane.

New Workspace VHDL Library - Creates a new workspace VHDL library. Shows the default library name in a text edit field in the Workspace pane.

New Project VHDL Library - Creates a new project VHDL library in the active project. Shows the default library name in a text edit field in the Workspace pane.

New Project Verilog Library Work - Creates a new VERILOG_WORK library in the active project.

New Project Verilog Library Other... - Opens a dialog for creating a new Verilog library in the active project.

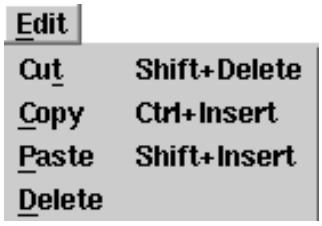
Edit Menu

Cut - Cuts the selection and moves it to the clipboard.

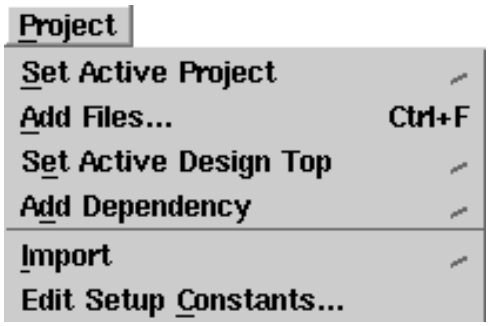
Copy - Copies the selection to the clipboard.

Paste - Pastes the clipboard to a designated area.

Delete - Deletes the selection.



Project Menu



Set Active Project - Sets the project selected from the submenu as the active project.

Add Files… - Displays the Add Files Dialog. Files are added to the active project.

Set Active Design Top - Sets the design top selected from the submenu as the active design top.

Add Dependency - Adds the project selected from the submenu to the active project.

Import Synopsys Setup File...- Displays the Open File dialog. The dialog only displays synopsys_sim.setup files. Use the dialog to import a synopsys_sim.setup file into the active project. Constants and libraries are added to the active project. (and to the synopsys_sim.setup file created by the Project window.) Examples are shown below.

synopsys_sim.setup contains:
WORK > RTL_LIB
RTL_LIB : ./work

ALU > ALU8
ALU8 : /tmp/ALU8
VERILOG < test.v
TIMEBASE = NS

Added to active project:
RTL_LIB: ./work   -Note WORK > RTL_LIB is ignored
ALU :/tmp/ALU8
ALU8 :/tmp/ALU8
VERILOG < test.v
TIMEBASE = NS

Import List of Files... - Displays the Open File dialog. Use the dialog to select a text file containing a list of Verilog or VHDL files. The files will be added to the active project, in the order that they appear. The list file may contain text comment lines that begin with '#' ,text lines that contain alpha/numeric names (which are ignored) that can optionally begin with '-', and text lines that contain file names (name.extension) separated by white space A text line may end with a backslash('\'). Examples are shown below.

#!/bin/csh -f   (Lines beginning with # are ignored)
vhdlan {vhdlan options} (name or -name is ignored)
file1.vhd
file2.VHD \
file3.vhd file4.vhd file5.vhd
file6.v file9.V \

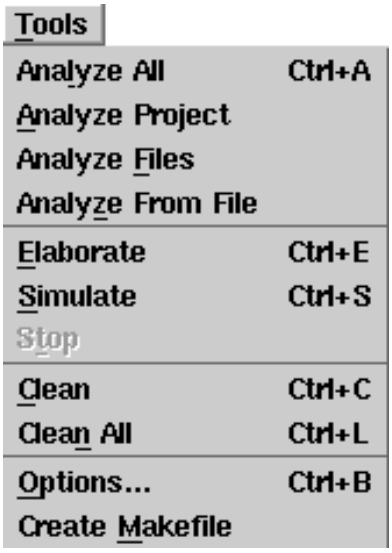
Edit Setup Constants - Opens the Edit Setup Constants dialog. Use this dialog to add, modify, or delete synopsys_sim.setup constants in the active project. The constants displayed in the dialog are added to the synopsys_sim.setup file created by the Project window when a project is analyzed, elaborated, or simulated.

Tools Menu

Analyze All - Unconditionally analyzes the active project and its dependent projects.

Analyze Project - Analyzes the active project and its dependent projects only if they are out of date. A project is out of date when the project library is cleaned, when source files are modified, when dependent projects are out of date, or when there are analyze errors.

Analyze Files - Analyzes the selected files in the Source File pane. The selected files must belong to the active project.



Analyze From File - Analyzes the files beginning with the selected file in the Source File pane. The selected files must belong to the active project.

Elaborate - Elaborates the active design top for the active project.

Simulate - Simulates the active design top for the active project.

Stop - Stops analysis or elaboration.

Clean - Cleans all libraries in active project. Does not clean dependent projects.

Clean All - Cleans all libraries in the active project and the dependent projects.

Options... - Displays the Options dialog.

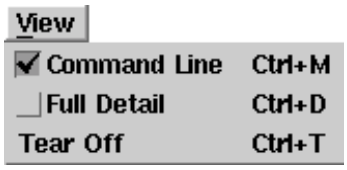
Create Makefile - Creates a makefile in the active project's directory. Use the makefile to analyze, elaborate, and simulate in batch mode. See Using a Makefile on page 9-56.

View Menu

Command Line - Shows or hides the command line edit window.

Full Detail - When checked, view pane shows all details.

Tear Off - The displayed view pane is displayed in a separate window.



Windows Menu

Hierarchy - Opens Hierarchy window.

Waveform - Opens Waveform window.

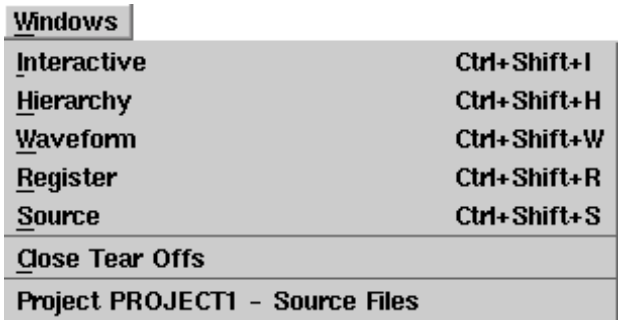
Register - Opens Register window.

Source - Opens Source window.

Interactive - Opens Interactive Invocation dialog.

Close Tear Offs - Closes all Tear Off windows.

List of Tear Off Windows - Moves selected Tear Off window to forefront.

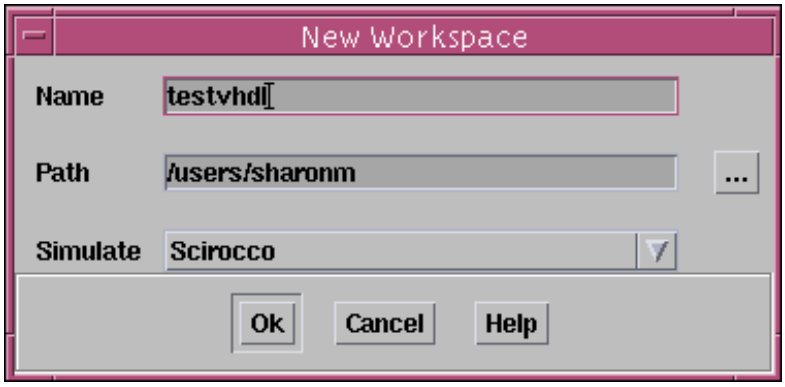


Dialogs

New Workspace Dialog

Use the New Workspace dialog to create a new workspace. A new project and a new project library named WORK are automatically created for the workspace. The new project name is displayed in a text edit field in the Workspace pane.

Figure 9-15 New Workspace Dialog



Name

Shows the name of the workspace.

Path

Shows the parent workspace directory.

... Select a Directory

Opens the Select Directory dialog. The selected directory is placed in the Path field.

Simulate

Shows the name of the simulator.

Ok

Creates a new workspace and closes the dialog.

Cancel

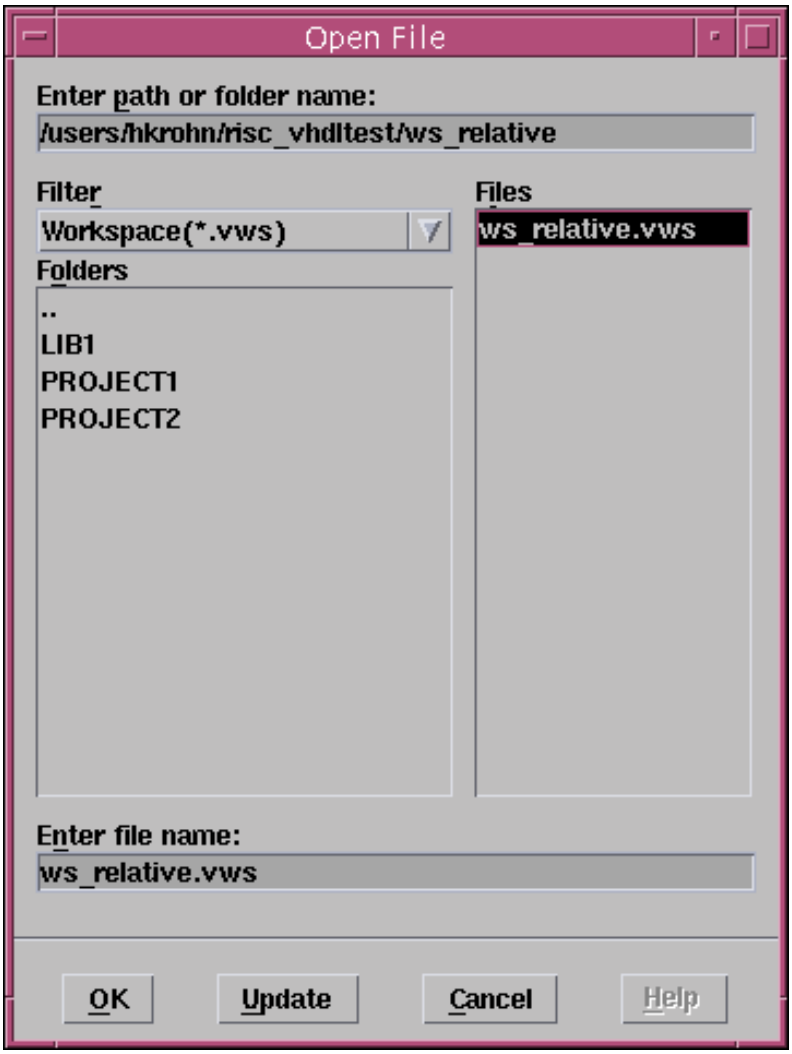
Closes the dialog and exits. Does not create a new workspace.

A new directory that has the workspace name is created in the ‘Path’ or parent workspace directory, if it does not already exist. The workspace file is placed in the workspace directory. The full name for the workspace file is /Path/workspaceName/workspaceName.vws. New project directories, files, and libraries are created in the workspace directory.

Open File Dialog

Use the Open File dialog to find a file to open. The dialog displays files that match the selected Filter.

Figure 9-16 Open File Dialog



Filter

Selects type of file to open.

Files

Displays list of files that match filter.

Folders

Lets you browse through the directory tree.

Ok

Opens file selected in “Enter file name” field.

Update

Updates Folders list.

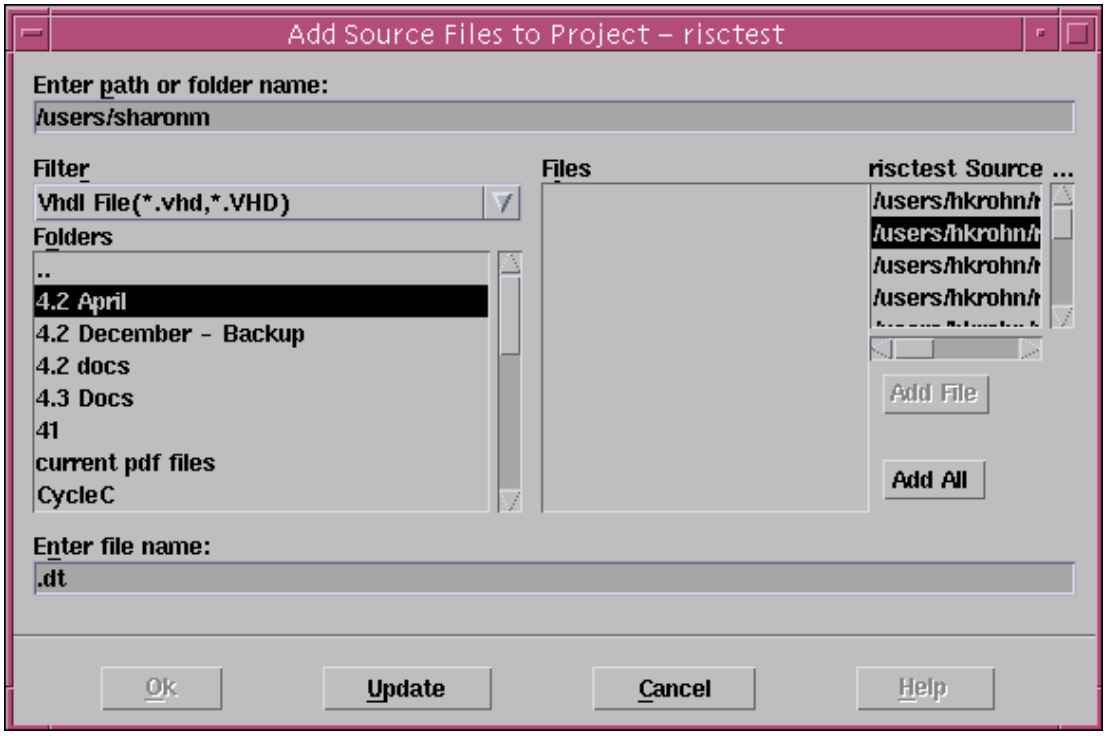
Cancel

Closes Dialog.

Add Files Dialog

Use the Add Files Dialog to add files to a project.

Figure 9-17 Add Files Dialog



Filter

Lets you choose a file filter. Displays valid VHDL and Verilog file extensions defined in BuildDialog file (see Files and Directories).

Files

Displays filtered files in Files list.

Add File

Adds the selected file to the active project.

Add All

Adds all Verilog and VHDL files in Files list to the project.

Folders

Use the Folders List to browse through the directory tree.

Ok

Adds the file to the project and closes dialog.

Update

Updates Folders list.

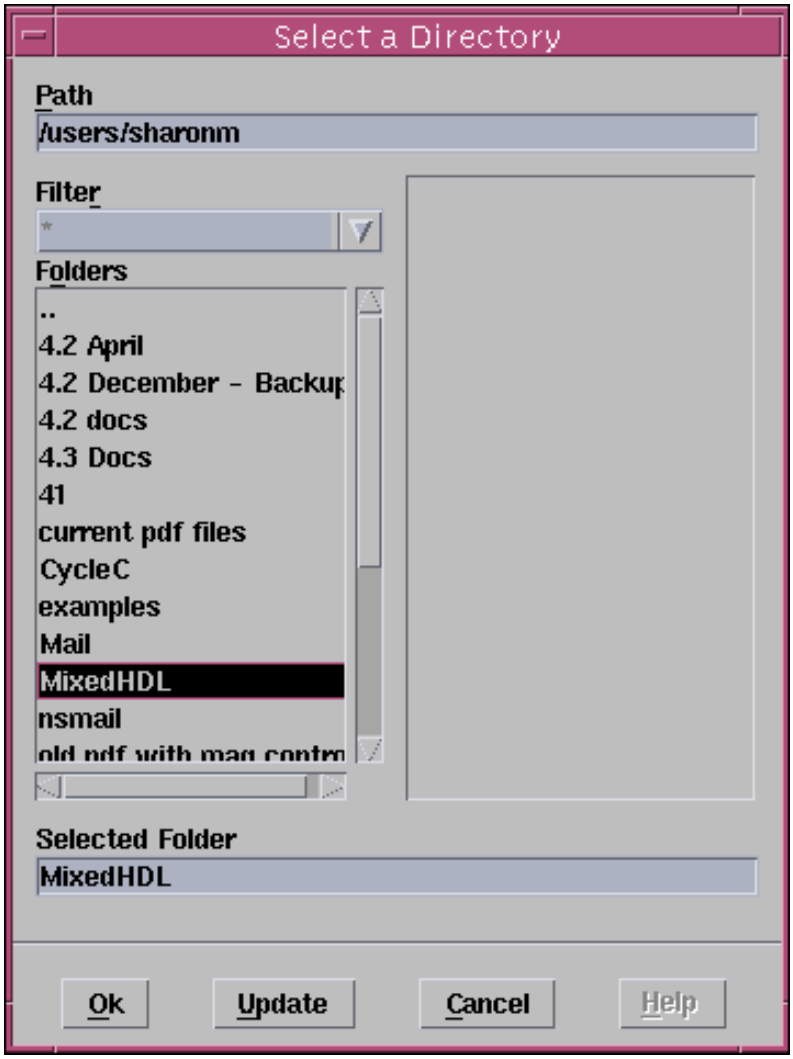
Cancel

Closes Dialog.

Select a Directory Dialog

Use the Select a Directory dialog to select a directory.

Figure 9-18 Select a Directory



Filter

Not Used.

Folders

Lets you browse through the directory tree. The selected directory is placed in the file name field.

Ok

Returns contents of file name field and closes dialog.

Update

Updates the Folders list.

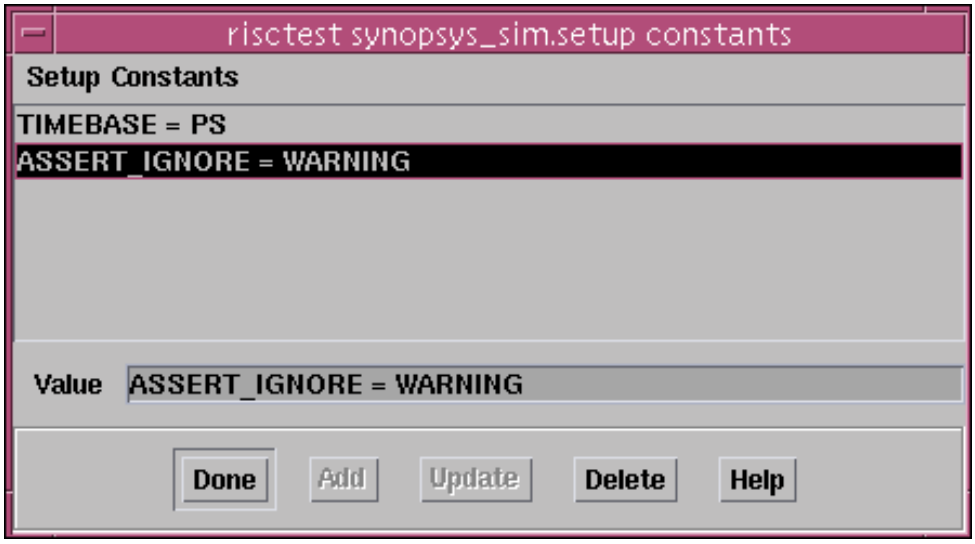
Cancel

Closes the Dialog.

Edit Setup Constants Dialog

The Edit Setup Constants Dialog is used to create, modify, and delete constants in the active project. The constants are written to the Project window synopsys_sim.setup file when the active project is analyzed, elaborated, or simulated.

Figure 9-19 Edit Setup Constants Dialog



Setup Constants

Displays list of constants in active project.

Name

Edit field for defining the name of a constant.

Value

Edit field for defining a constant value.

Done

Closes the dialog.

Add

Adds the constant in the Name edit field.

Update

Changes the value of the selected constant.

Delete

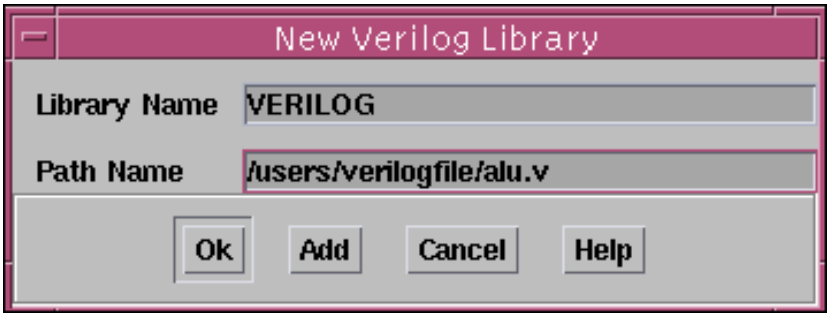
Deletes selected constant.

New Verilog Library Dialog

The New Verilog Library dialog is use to create new Verilog libraries.

Verilog libraries have the form ’name < pathname’ in the synopsys_sim.setup file.

Figure 9-20 New Verilog Library Dialog



Library Name

Specifies the logical name of the Verilog library

Path Name

Specifies the pathname of the Verilog library file or directory. If the Library is a library directory, after the path name enter a colon (:) followed by one or more filename extensions to specify the extensions of the files in the library that you want searched (to resolve module instances). For example:

```
VLOGLIB1 < /net/cores/vloglib1:.v+.vp
```

Ok

Creates the new library entry in the synopsys_sim.setup file and closes the dialog.

Add

Creates the new library entry in the synopsys_sim.setup file and clears the fields so you can specify more libraries.

Cancel

Closes the Dialog.

Help

Displays an menu page for help topics on Verilog Libraries

Using the Options Dialog

A project can be analyzed, elaborated, and simulated using default command line settings. The Options dialog is used to change the default settings.

The Options Dialog always displays the Analyze, Elaborate, and Simulate tab panes. Both the Verilog Compile tab and Verilog Run tab are displayed when a workspace contains Verilog files.

The left hand Workspace view pane displays objects whose options can be changed. The objects displayed under Project Folders (source files and design tops) change when the tab is changed.

Options Dialog Buttons

Ok

Sets options on selected objects and closes the dialog.

Apply

Sets options on selected objects.

Apply Defaults

Sets selected tab pane to default values. Use Ok or Apply to set options for selected Workspace view objects.

Cancel

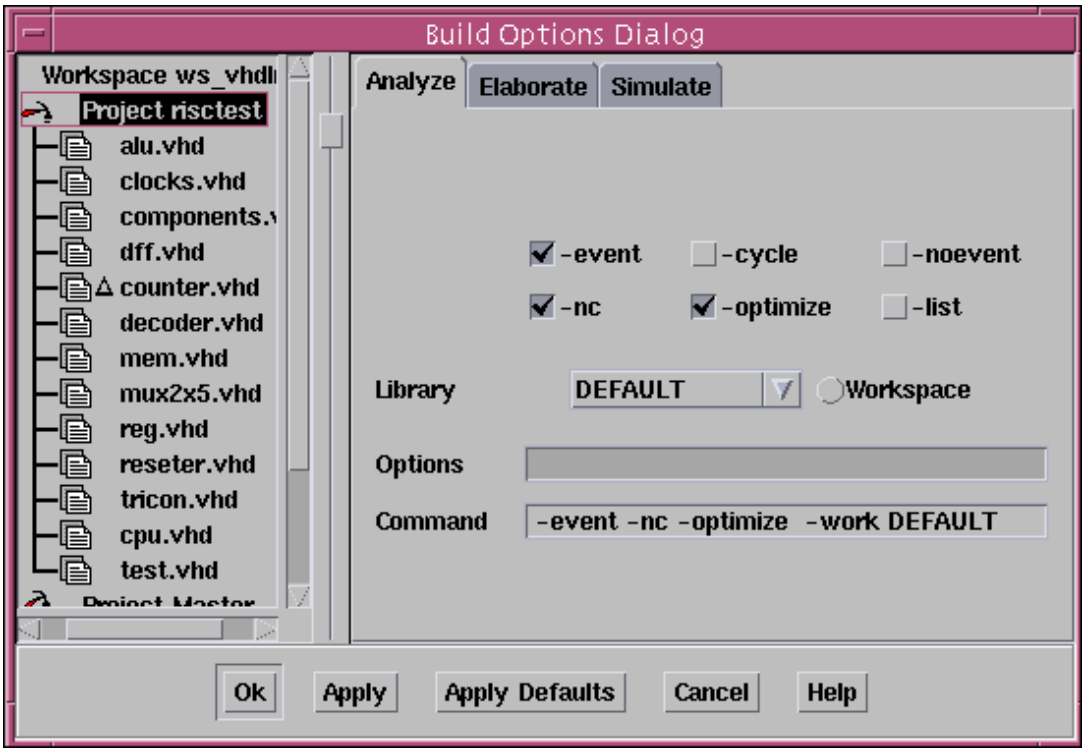
Closes the dialog.

Help

Displays help information.

Setting Analyze Options

Figure 9-21 Analyze Options



Use the Analyze tab to set analyze options for a workspace, project, or VHDL source file.

Options are set using check buttons, the Library combo box, and the Options edit field. The Command field displays options that are passed to the analyzer.

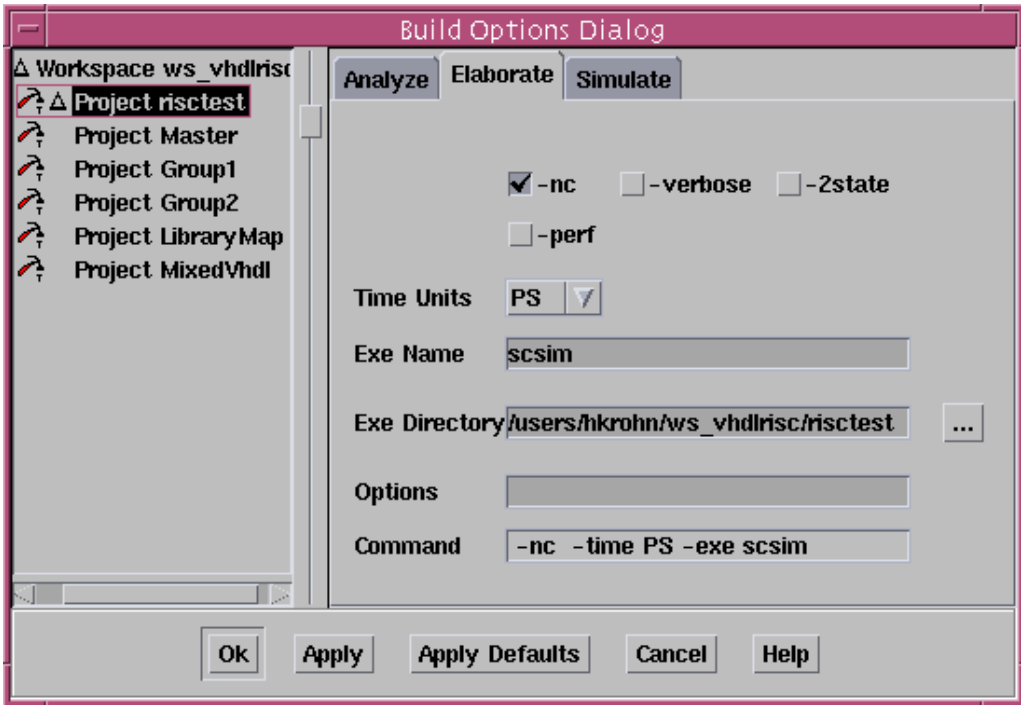
The Library combo box is used to select a library that receives analyzer output. Use the radio button to choose a workspace library or a project library.Analyzer output will go the active project library until a library is specified with this combo box.

Options can be modified for the workspace, projects, or VHDL source files. Separate sets of options are maintained for the workspace, project and files. Initially the three sets of options are identical.

Analyze settings for a VHDL source file are determined as follows: Use the file options if they are changed, otherwise use the project options if they are changed, otherwise use the workspace options. A black triangle icon indicates that options are changed.

Setting Elaborate Options

Figure 9-22 Elaborate Options



Use the Elaborate tab to set elaborate options for a workspace, project, or design top.

Options are set using check buttons, the Time Units combo box, the Name edit field, the Directory edit field and the Options edit field. The Command field displays options that are passed to the analyzer.

The Name Edit field is used to name the simulation executable.

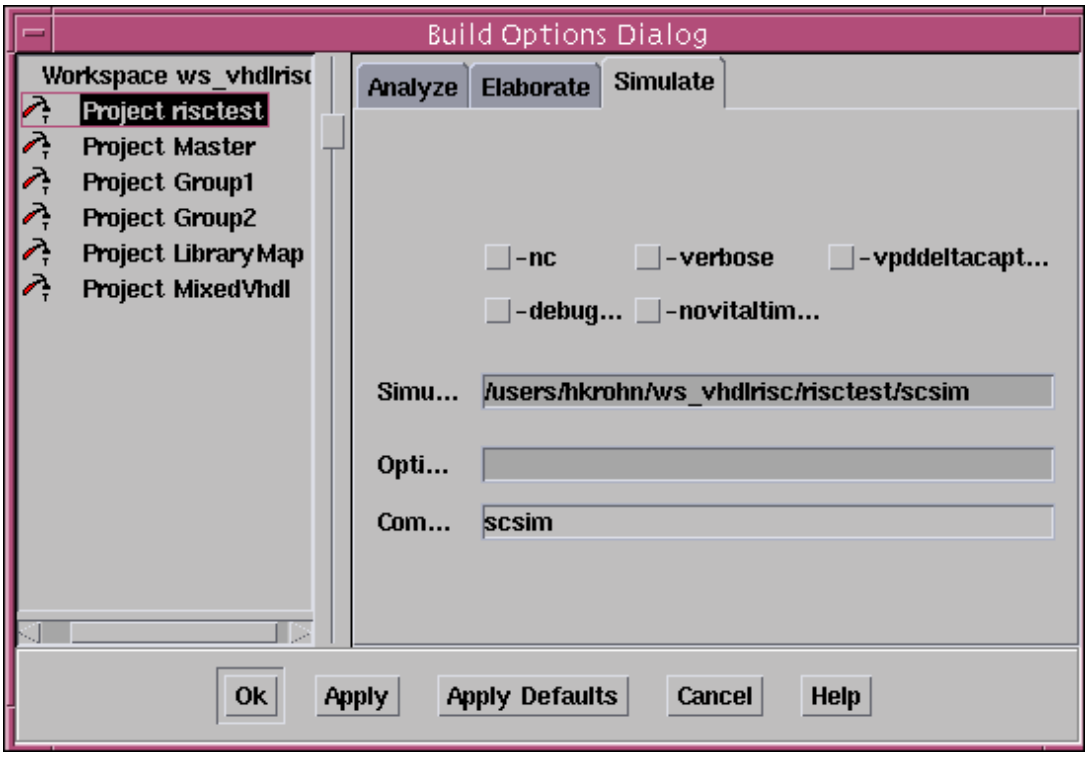
The Directory edit field contains the simulator directory. The elaborator places the simulator in this directory. The interactive VCD+ history file is found in this directory, after the simulator is executed. If a VHDL design contains a file that is loaded during simulation, change the contents of the Directory field to the directory containing the file. The button next to the directory edit field can be used to open the “Select a Directory Dialog”.

Options can be modified for the workspace, projects, or design tops. Separate sets of options are maintained for the workspace, project, and design tops. Initially the three sets of options are identical.

Elaborate settings for a design top are determined as follows: use the design top options if they are changed otherwise use the project options if they are changed otherwise use the workspace options. A black triangular icon indicates that options are changed.

Setting Simulate Options

Figure 9-23 Simulate Options



Use the Simulate tab to set runtime simulator options for a workspace, project or design top.

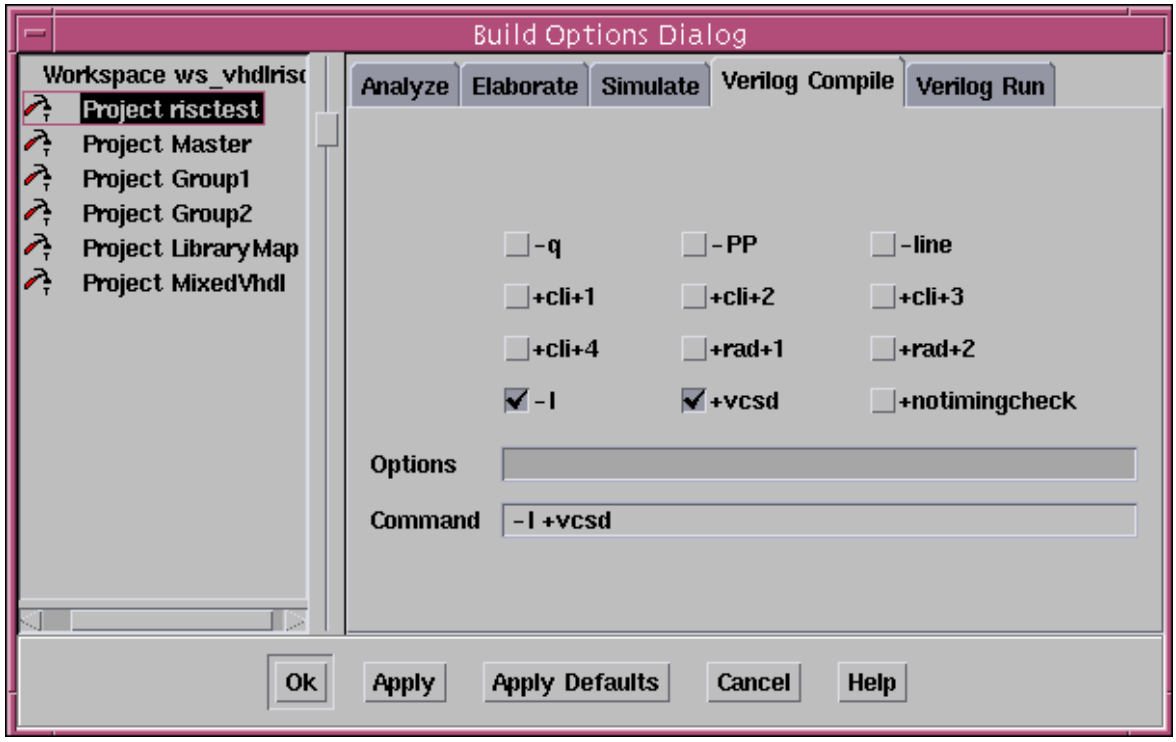
Options are set using check buttons and the Options edit field. The Command field displays options that are passed to the analyzer. The Command field also displays the name of the simulation executable, from the elaborate tab.

Options can be modified for the workspace, projects, or design top. Separate sets of options are maintained for the workspace, project and design top. Initially the three sets of options are identical.

Simulate runtime settings for a design top are determined as follows: Use the design top options if they are changed otherwise use the project options if they are changed otherwise use the workspace options. A black triangular icon indicates that options are changed.

Setting Verilog Compile Options

Figure 9-24 Verilog Compile Options



Use the Compile tab to set VCS compile-time options. These options are passed to the elaborator as a string using the `-verilogcomp` option.

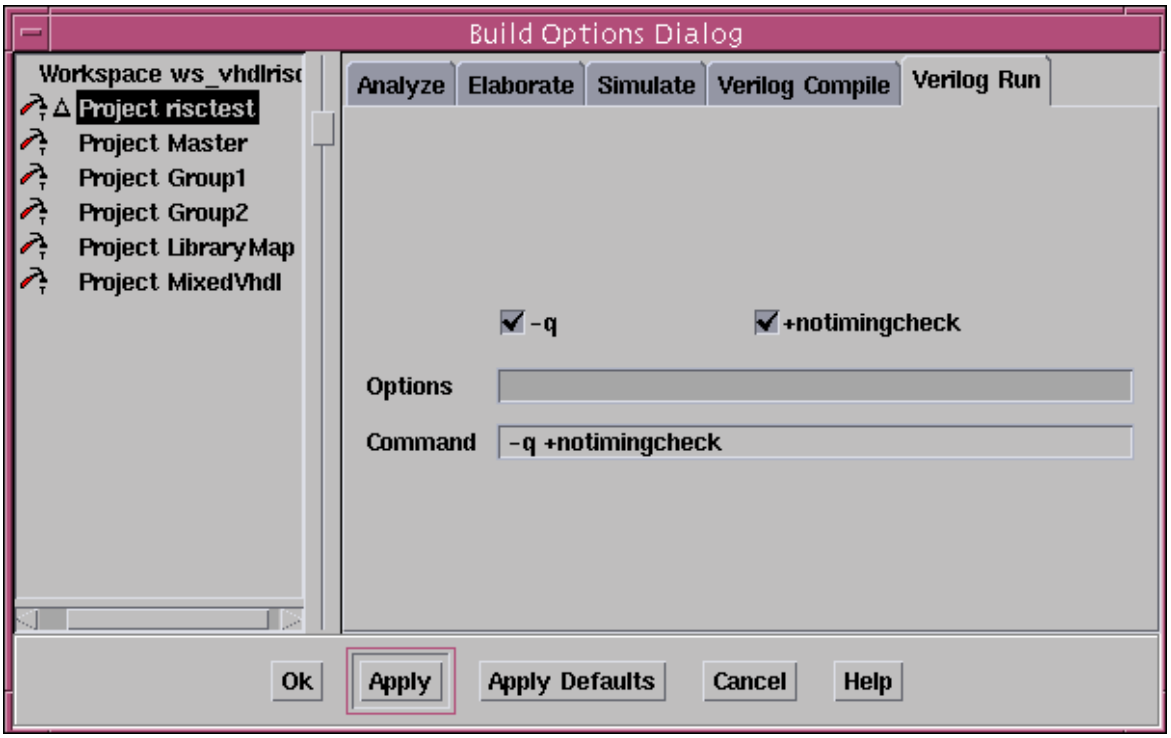
Options are set using check buttons and the Options edit field. The Command field displays options that are passed to the Verilog compiler.

Options can be modified for the workspace or project. Separate sets of options are maintained for the workspace. Initially the two sets of options are identical.

Compile settings for a Verilog project are determined as follows: use the project options if they are changed, otherwise use the workspace options. A black triangular icon indicates that options are changed.

Setting Verilog Run Options

Figure 9-25 Verilog Run Options



Use the Verilog Run tab to set VCS runtime options. These options are passed to VCS as a string using the `-verilogrun` option.

Options are set using the check buttons and the Options edit field. The Command field displays the options that are passed to VCS.

Options can be modified for a workspace, project, or a design top. You can maintain separate sets of options for different workspaces, projects, and design tops. Initially the three sets of options are identical.

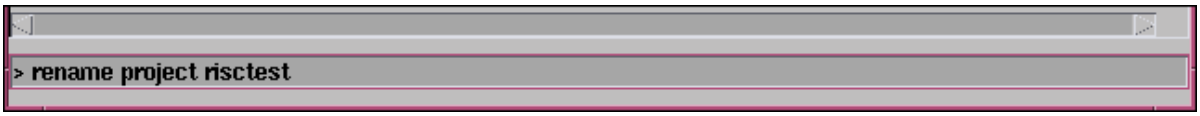
VCS runtime options for a design top are determined as follows: use the design top options if they have been changed, otherwise use the project options if they have been changed, otherwise use the workspace options. A black triangle icon indicates that options have been changed.

Executing Commands from the Command Line

The command line is used to execute commands and scripts without using the GUI. At the command line prompt, enter a command and press the Enter key. The command is executed and the History pane displays the command and any error messages.

Commands executed from the command line are placed in a history buffer. Use the up/down arrow keys to move commands from the command buffer to the command line edit field.

Figure 9-26 Command Line



Commands

The Project Window commands can be executed from the GUI, the command line, or placed in a file and then executed.

Definitions

- filename – quoted, full path name of a file. For example “/tmp/alu.vhd”.
- *librarydir*, *workspacedir* – quoted, full path name of a directory. For example “/tmp/test”.
- [ ] – fields inside brackets are optional.
- {option | option1 | …} use one of the list of options.
- **library** *libname* [**project** *projname*] specifies a workspace library unless the **project** option is used, then it specifies a project library.

The Project Window commands are as follows:

active library *libname*

Sets ’*libname*’ as the default library for the active project.

activate project *projname*

Sets ‘*projname*’ as the active project.

activate top *topname* **library** *libname* **[project** *projname***]**

Sets ‘*topname*’ as the active design top. ‘*topname*’ is found in library *libname*.

add [before | after *filename***] file** *filename1*

Adds source file ‘*filename1*’ to the active project. The command has an option to place the file before or after ‘*filename*’.

add constant *string*

Add constant to the active project.

add top *topname* **library** *libname* **[project** *projname***]**

Adds design unit ‘topname’ to the active projects list of design tops.

add [before | after *projname***] dependent** *projname1*

Adds project ‘*projname1*’ to the active projects dependency list. The command has an option to place the project **before** or **after** ‘*projname*’.

add filelist *filename*

File *filename* contains a list of Verilog or VHDL source files that are added to the active project.

add setup *setupdir*

Directory *setupdir* contains a synopsys_sim.setup file that is read into the active project. Adds constants, and both VHDL and Verilog libraries to the project.

add constant *name value*

Adds a constant and value to the active project. The constant is written to the synopsys_sim.setup file created by the Project window.

analyze project

Analyzes active project and each dependent project only if each project needs to be analyzed.

analyze all

Unconditionally analyzes the active project and all dependent projects.

analyze from *filename*

Analyzes the active projects list of source files beginning with '*filename*'.

analyze file *filename* [*filename1* [*filename2*…]]

Analyzes the list of files for the active project.

clean [all]

Cleans the active projects libraries. The **all** option causes all dependent project libraries to be cleaned, in addition to the active project. All files in a library directory are removed, when a library is cleaned.

clean {workspace | project} *libname*

Cleans the indicated library.

close workspace

Closes the open workspace.

delete constant *string*

Deletes the constant from the active project

delete file *filename* **[***filename1* **[***filename2***..]]**

Removes the list of files from the active project.

delete top *topname* **library** *libname* **[project** *projname***]**

Removes the design top from the active project.

delete project *projname*

Deletes '*projname*' from the workspace.

delete dependent *projname*

Removes '*projname*' from the active projects dependency list.

delete library {workspace | project} *libname*

Deletes ‘*libname*’ from the workspace or project.

delete library verilog *libname*

Deletes Verilog library **libname** from the active project.

delete constant *name*

Deletes a constant from the active project.

elaborate

Elaborates the active design top.

exit

Exits VirSim.

move {before | after} *filename* **file** *filename1*
move {up | down} file *filename1*
move {before | after} *projname* **dependent** *projname1*
move {up | down} dependent projname1

Moves the position of file ‘*filename1*’ or dependent project ‘*proname1*’ in the active project.

new workspace *workspacedir* **[***simulatorname***]**

Creates a new workspace. The workspace directory ‘*workspacedir*’ is created if it does not already exist. The name of the workspace simulator is defined by *simulatorname*. The default simulator is Scirocco.

new project *projname*

Creates a new project ‘*projname*’ and a project library DEFAULT. The parent directory for the library is the project directory.

new library {workspace | project} *libname* **[***librarydir***]**

Creates a new workspace or project library ‘*libname*’. If directory ‘*librarydir*’ is not specified, the library is created in the workspace or project directory using *libname* as the directory name.

new library verilog *libname pathname*

Creates a new workspace or project library ’*libname*.’ THe pathname can be either a quoted string or a character string.

open workspace *filename*

Opens workspace ‘*filename*’.

rename project *projname*

Changes the name of the active project to ‘*projname*’.

rename library {workspace | project} *oldname newname* **[***librarydir***]**

Renames a workspace or project library. To change ‘*librarydir*’ without changing the library name, set ‘*newname*’ equal to ‘*oldname*’. Note that like the **new library** command, ‘*librarydir*’ can be an empty string.

rename library verilog *oldname newname pathname*

Renames a project Verilog library. To change the *pathname* without changing the library name, set ’*newname*’ to ’*oldname*.’

save workspace

Saves the workspace creating new *.vws and *.vpj files.

save makefile

Saves a makefile in the active project directory.

set *option_type* **{workspace | project** *projname***}** *string*
set *option_type* **file** *filename string*
set *option_type* **top** *topname string* **library** *libname* **[project** *projname***]**

The **set** command uses the following definitions.

option_type = {**analyze** | **elaborate** | **verilog compile** | **verilog run**}

string = quoted string of options (“-time PS –directory /users/tmp”)

You use this command to set analyze, elaborate, compile, and simulate options for the active project or workspace. Examples of these commands can be found in workspace (.vws) and project (.vpj) files when you use the Options dialog to set options for workspace, projects, files and design tops.

simulate

Simulates the active design top.

source *filename*

Executes commands in ‘*filename*’.

stop

Stops elaborating or analyzing the design.

Files and Directories

When the Workspace dialog is used to create a workspace, three new directories are created if they do not already exist, the workspace directory ./workspaceName, the project directory ./workspaceName/PROJECT1 and a project library ./workspaceName/PROJECT1/WORK. Saving the new workspace creates two new files, workspaceName.vws, which is in the workspace directory and PROJECT1.vpj, which is in the project directory. The vws and vpj files contain commands that restore the workspace when it is opened.

The Project Window creates files in a project directory when a project is analyzed. The analyzeHistory file contains analyze history information. A sysnopsy_sim.setup file is created when source files are analyzed.

A synopsys_sim.setup file is created in the simulation directory (see Setting Elaborate Options on page 9-42), when elaborating or simulating a design.

The Project Window creates a directory called .virsim in your home directory if it does not already exist. The directory contains files used by the Project Window.

File ./virsim/projectBrowser contains configuration information and it must not be modified. VirSim creates the file if it does not exist.

File BuildDialog is in VIRSIMHOME/appfiles. This file defines Options dialog buttons and can be modified. The file also contains valid Verilog and VHDL file extensions. You can place a modified copy of this file in .virsim directory. The Project Window first looks for this file in the .virsim directory and then in the VIRSIMHOME/appfiles directory.

By default, the interactive simulation file is placed in the project directory, unless you use the Options dialog to specify a different simulation directory.

Moving Projects

Saved paths in a workspace are relative to the workspace path. Thus, a user can easily move a workspace.

For example, assume the workspace path is `/users/hkrohn/risc` and workspace name is `ws_relative`. The Project Window creates a new directory `ws_relative` and saves the workspace in `/users/hkrohn/risc/ws_relative/ws_relative.vws`. A new project directory is also created called `/users/hkrohn/risc/ws_relative/PROJECT1`. The project is saved as `./ws_relative/PROJECT1/PROJECT1.vpj`.

If a source file path is `/users/hkrohn/risc/alu.vhd`, then the saved path will be `./alu.vhd` otherwise the complete path is saved. This makes it possible to move(copy) the workspace directory to a new location.

To move the workspace when source files are not in the workspace path: `cd /users/hkrohn/risc ; mv ws_relative newlocation;`. To move the workspace and source files, `cd /users/hkrohn ; mv risc newlocation ;`.

Using a Makefile

To create a makefile for a project, make sure the project is active then select Create Makefile from the Tools menu. The makefile has four targets: all, analyze, elab and sim. When the makefile is created, a synopsys_sim.setup file is written to the workspace path, overwriting any existing setup file.

To run the makefile, go to the parent workspace directory and execute the makefile that is contained in the project directory. For example, if the workspace parent directory is `/users/hkrohn/risc`, do the following:

```
cd /users/hkrohn/risc
make -f ws_relative/PROJECT1/Makefile analyze
```

10

Time Units

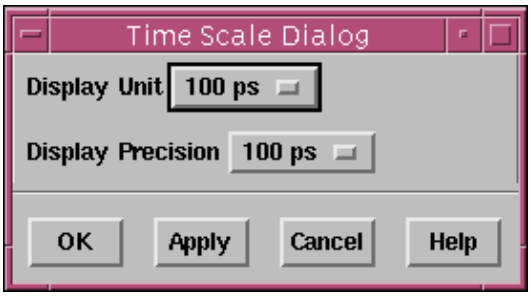
This chapter describes how to set display unit and precision values for Time Units.

The Time Scale Dialog allows you to configure display units and precision. Because the use of multiple history files may involve different units and precision, it may be necessary to configure the time scale.

The Time Scale Dialog has two option menus (Display Unit and Display Precision) and four action buttons.

See Figure 10-1, Time Scale Dialog.

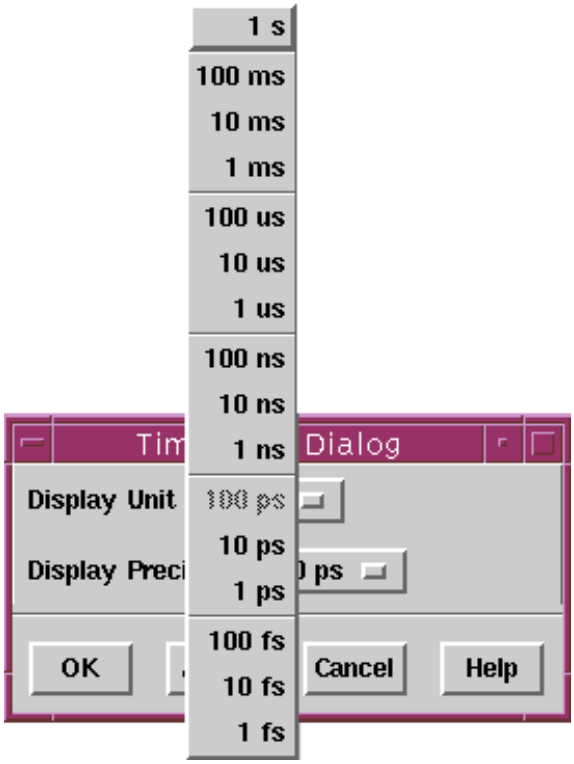
Figure 10-1 Time Scale Dialog



Display Unit

Opens an option of valid time units (see Figure 10-2, Time Selection Menu). Click left on the Display Unit button to change the units that VirSim uses to display time. In the popup menu, click left on the desired time unit. The new time appears on the button face.

Figure 10-2 Time Selection Menu



Display Precision (Button opens option menu)

Opens an option menu of time precision values. From the popup menu, click left on the desired time precision selection. The new time appears on the button face.

OK

Changes time values to the unit and precision selected and closes the Time Scale Dialog.

Apply

Changes time values to the unit and precision selected and leaves the Time Scale Dialog open.

Cancel

Closes the Time Scale Dialog without saving any changes.

Help

Opens online help for the Time Scale Dialog.

Note:

Changes in display unit and precision are not stored in configuration files.

11

Radices

This chapter describes the use of the Radix Dialog to create, modify and delete custom radices. A custom radix is used to map vector values to user-defined text strings. For example, the logic value of 6'h04 may be mapped to a display string of HLT. By setting a signal's radix (for example, in the Waveform Window Signal Value pane) to the new radix, all values of 6'h04 are displayed as HLT.

Using a Radix

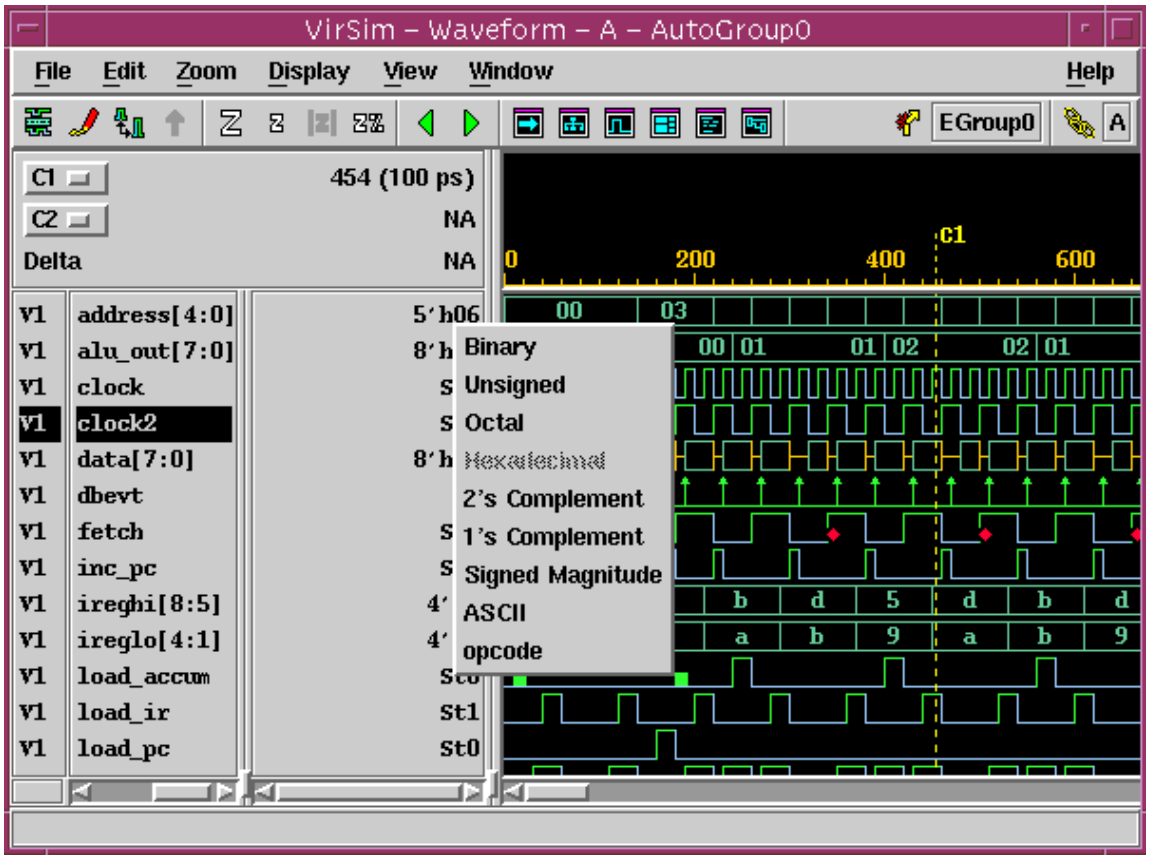
Use the following procedure to change the radix of a signal displayed in a VirSim window.

1. Point the cursor over the value for a signal or group of selected signals and click right. A popup menu appears with a list of the standard and user-defined radices, (see Figure 11-1, Radix List).

2. Position the mouse over the desired radix and release the mouse button.

   For radix value strings, each character represents one bit for binary, 3 bits for octal, and 4 bits for hex. Values are grouped from right to left (e.g. 6'b1xzzzz would be 6'hXz where X=1x and z=zzzz). For decimal values you will either get one non-numeric character or a decimal number representing all the bits.

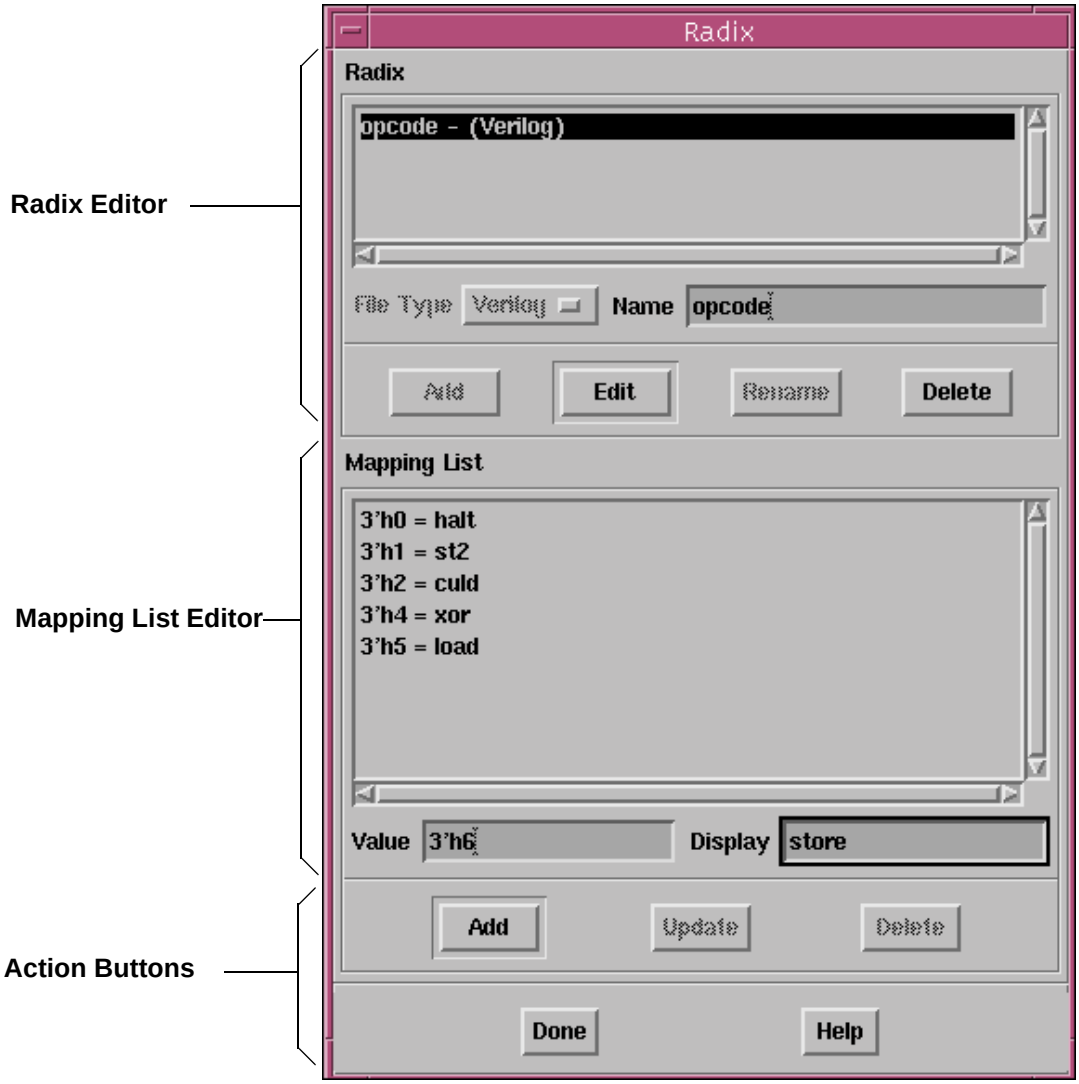
Figure 11-1 Radix List



Radix Dialog

(Verilog only) Figure 11-2, Sample Radix Dialog shows the major areas of the Radix Dialog.

Figure 11-2 Sample Radix Dialog



Radix Editor

(Verilog only) The Radix Editor is used to create new Radices, edit the mappings of a Radix, rename a Radix, and delete a Radix.

Radix List

Displays all Radices that have been entered. Selecting a Radix Name in this area, then clicking **Edit** fills in all fields of the dialog with the values associated with this Radix and allows editing of the Mapping List for this Radix. A radix must be added to the Radix List with the Radix **Add** button before it can be edited with the mapping editor.

File Type

Select the file type that can use this radix. Not all file types can use user-defined radix features.

Name

Used to enter a Radix Name. Radix Names may only contain alphanumeric characters and underscores and can begin with either an alpha character or underscore.

Mapping List Editor

(Verilog only) The Mapping List Editor is used to create, update, and delete a Radix's mapping of values to text strings.

Mapping List

Displays all mappings that have been entered. Selecting a mapping in this area fills in the fields of the dialog with the values associated with this mapping and allows editing of the mapping.

Value

Used to enter the value to be mapped. The value uses the Verilog number format and can be entered in binary, decimal, octal or hexadecimal. Legal values include `-15`, `4'hf`, `'o777` and `17'd9`. In addition, the wildcard `?` can be used to indicate that any single digit is allowed in the wildcard position. Examples of this are `16'hfaa?` and `4'b000?`. The wildcard `?` cannot be used in decimal values.

Display

Used to enter the text string to be displayed in place of the actual value. Any alphanumeric character is allowed.

Creating a User-Defined Radix

(Verilog only) Use the following procedure to create a user-defined radix.

1. Click left in the **Name** field and type a radix name.
2. After the radix name, press **Enter** to add the name and enable Edit.
3. Click left on **Edit** or press **Enter** to edit the value mappings for the new Radix.
4. Type a value to be mapped and press **Enter**.
5. Type the text string to be displayed for the previously entered value and click left on **Add** or press **Enter**.
6. To enter additional mappings, repeat step 3 and 4 and then click left on **Done** after the last mapping.

12

Event Origin

The VirSim Event Origin feature (Verilog only) can help you to obtain precise information about the origin of a signal value change at a specific time. VirSim can display signals from all levels of a design hierarchy, including signals named differently but logically equivalent by connection through a module port. Whatever the signal data type and location in the hierarchy, there is a root cause for every signal change, referred to as its event origin.

Before Using Event Origin

Event Origin requires that VirSim compile the Verilog source so it can obtain information about the design that is not available from the simulator or vpd file. Source and compile options used must match those that were used when the simulation was run. If the Verilog code has not already been compiled, you will be prompted for the source by the Load Sources Dialog box when Event Origin is selected.

If line trace data was saved in the VCD+ file, the Event Origin function will point to the line of source that caused the transition. It also uses this data to eliminate possible drivers that were not actually executed at the time the event occurred.

Note:

Older VCD+ files are missing some information to allow driver elimination. To use this feature you should create your VCD+ files using a VirSim 3.0 or later PLI.

Where You Can Use Event Origin

Event Origin is allowed at various places in the different windows. When event origin is not allowed it is not sensitive, which is due to one of the following:

1. The window is linked to SIM. Event origin is not allowed when the window is linked to a simulator. Unlink from SIM and then try event origin.
2. A single net can not be determined from the current mouse location (expressions are not allowed).

The following is a list of objects for each window where event origin is allowed:

- Waveform Window: Any waveform.
- Register Window: Any signal.
- Source Window: Any valid signal name.
- Logic Browser: Any port, port instance, primitive terminal, or focus net.

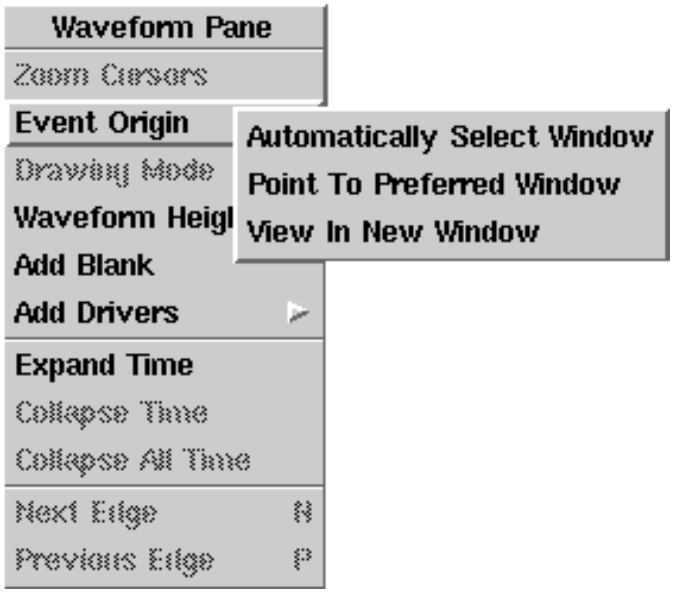
The starting time is the current time displayed for the given window unless you are in the Waveform Window. In this case, the time is determined by the position of the mouse cursor. The event time is found by looking for the first change that occurred at or before the starting time.

Event Origin works on vectors as well as scalars. For vectors, only the bits that changed at the event time are considered.

How to Use Event Origin

To use event origin place the cursor on the object for which you wish to see the origin(s), then right click to bring up the context sensitive menu. Select Event Origin from the menu. This brings up a sub menu of Event Origin options.

Figure 12-1 Event Origin Menu via the Context Sensitive Menu



Automatically Select Window

This option will select an existing window or create a new window. Primitive terminals and continuous assignments are displayed in a Logic Browser window. Procedural assignments are displayed in a Source Window. If this window is already open it will be used to display the Event Origin. Otherwise, a new one will be created and linked to the one you are in.

Point to Preferred Window

This option allows you to select either a Source Window or Logic Browser to display the driver. When this option is selected, the cursor will change to a left pointing hand. Left click in the desired Source or Logic Browser window and the selected window displays the event origin(s). (If you do not click in a valid window, Event Origin terminates, and VirSim returns to normal processing.)

View In New Window

This option will always create a new window. The type of window is determined as described in the Automatically Select Window option. If possible the new window will be linked to the same link group as the window where this option was selected. Otherwise the window will get a new link group.

Display the Event Origin

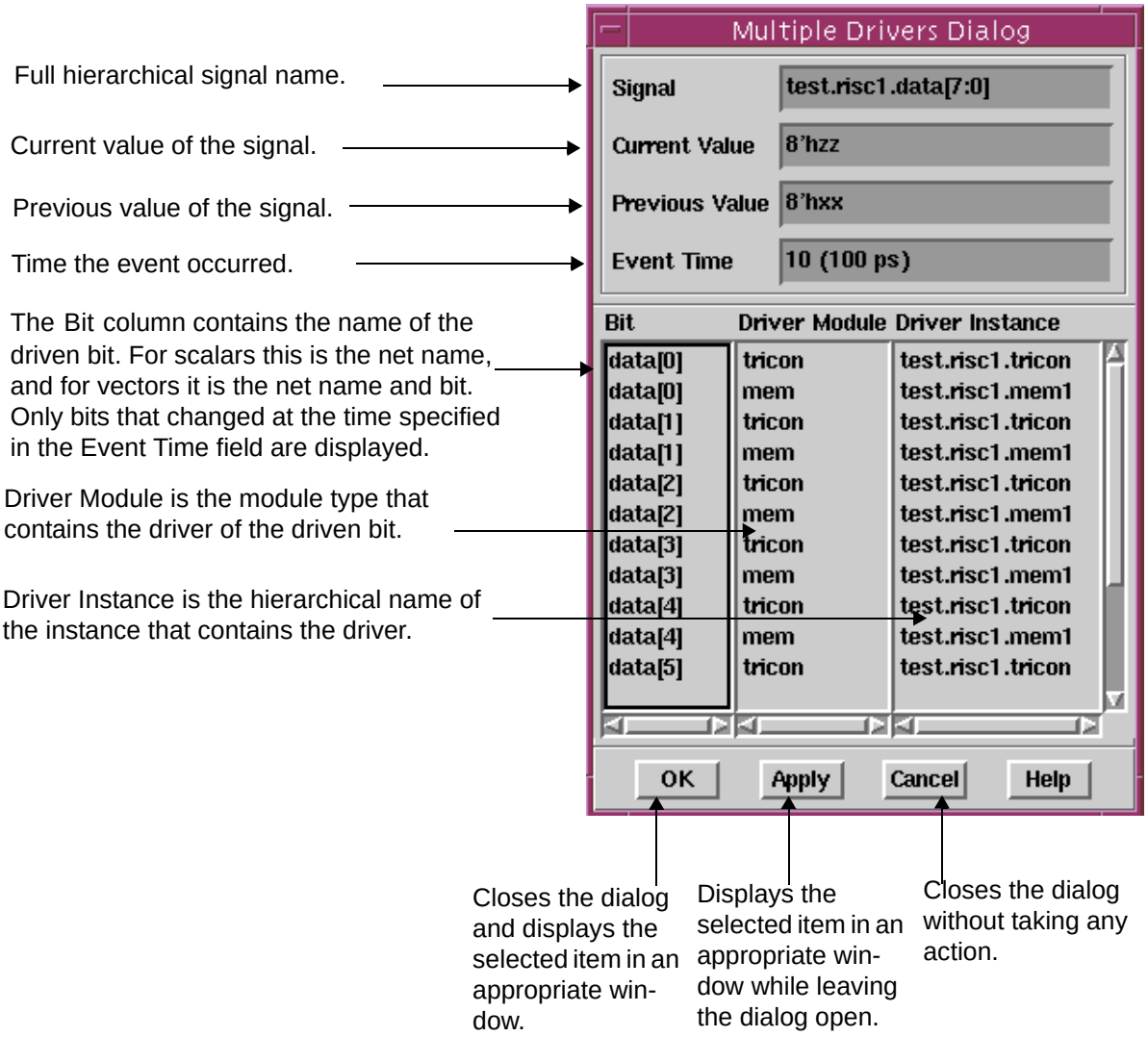
When the event origin is displayed in a Source Window, the instance containing the origin is placed in the EventOrigin instance group. The window is then changed to use the EventOrigin instance group and display the proper source. The line representing the driver is marked with an off-white outline arrow. The arrow is filled in with yellow if line data was saved and the line data was saved for the driving signal. This indicates that you can step forward or backward from the point where this line was executed. If the arrow is hollow, line stepping occurs at the beginning of the time when the event occurred. All the other source window functions are still available to help you analyze why a particular change occurred.

When displayed in the Logic Browser, the net being driven by the event origin is displayed and the time is set to the beginning of the time the event occurred. You can then use the normal logic browser navigation and change search features to further analyze the problem.

Choose from Multiple Drivers for an Event

If you invoke the event origin function and there is more than one driver contributing to the event, the Multiple Drivers Dialog appears. This dialog shows all drivers contributing to an event and allows you to select an item for display. To display a driver, you can double left-click on any of the three items for that driver, or you can single left-click to select an item and use the **OK** or **Apply** buttons.

Figure 12-2 Multiple Drivers Dialog


Full hierarchical signal name.
Current value of the signal.
Previous value of the signal.
Time the event occurred.
The Bit column contains the name of the driven bit. For scalars this is the net name, and for vectors it is the net name and bit. Only bits that changed at the time specified in the Event Time field are displayed.
Driver Module is the module type that contains the driver of the driven bit.
Driver Instance is the hierarchical name of the instance that contains the driver.
Multiple Drivers Dialog
Signal
test.risc1.data[7:0]
Current Value
8'hzz
Previous Value
8'hxx
Event Time
10 (100 ps)
Bit
Driver Module
Driver Instance
data[0] tricon test.risc1.tricon
data[0] mem test.risc1.mem1
data[1] tricon test.risc1.tricon
data[1] mem test.risc1.mem1
data[2] tricon test.risc1.tricon
data[2] mem test.risc1.mem1
data[3] tricon test.risc1.tricon
data[3] mem test.risc1.mem1
data[4] tricon test.risc1.tricon
data[4] mem test.risc1.mem1
data[5] tricon test.risc1.tricon
OK
Apply
Cancel
Help
Closes the dialog and displays the selected item in an appropriate window.
Displays the selected item in an appropriate window while leaving the dialog open.
Closes the dialog without taking any action.

Event Origin Classifications

The event origin depends on the type of signal, which can be classified into one of three major categories:

1. Register (`reg`, `integer`, `time` and `real`) variables, where the values are assigned by procedural statements within initial blocks, tasks, and functions. For these variables, the event origin is defined as the procedural statement that last modified the variable at the event time. These statements often exist in the same scope as the variable declaration, but need not, as in the case of a statement that assigns to a variable by hierarchical reference (For example, `top.cpu.PC = 32'h0000ff00`).

2. Net (`wire`, `tri`, `or`, `wor`, `and`, `wand`, `trireg`, `tri1`, and `tri0`) variables, which derive values from one or more continuous assign statements, primitive terminal connections, or register variables, drive values onto the net through port connections. The drivers can exist in the same scope as the net, but can just as easily exist in a different scope in the hierarchy, as is often the case of a net connected to a module port instance.

3. Signals representing the output of a single primitive to a multi-driver net. Such signals do not correspond to a variable in the HDL source, but are produced by the VirSim VCD+ writer during simulation to allow more accurate analysis of multi-driver nets. The origin in this case is simply the primitive that produced the contribution. These signals are only produced when the **+vpddrivers** argument is given to the simulator.

Debug with Event Origin

For a register variable (`reg, integer, time`, or `real`), a value change always results from the execution of a procedural statement. For a signal declared as a net data type (`wire`, `tri1`, etc.), a value change usually results from a change in the output of a driving primitive gate or continuous assign statement (often in a different scope of the hierarchy from the net declaration itself), or may result from a procedural force statement.

Since force statements are done outside the logic of the HDL source code, doing an Event Origin on a force statement may give misleading results. If line trace information was not saved, all the drivers of the affected net will be listed but may not be the cause of the change if the force occurred after the driver was executed. If line trace information was saved there are two possible outcomes.

1. A driver was executed at the same time the force was entered. This case is similar to when line trace data was not saved. The driver(s) will be displayed but if the force was entered after a driver executed, that driver would not be the cause of the change.
2. No drivers were executed at the time the force was entered. VirSim will display the Event Origin.

For a net driven by a single primitive or continuous assign (with the exception of a procedural force to the net), the event origin is fixed.

For nets and register variables with multiple drivers, the event origin may change. One or more drivers may be responsible for a particular net value change, and the set of responsible drivers varies. Since the drivers may reside in any scope. It can be problematic to establish the underlying cause of the net value change.

Using the **+vpddrivers** option can aid in debugging problems with multiple drivers since you can see the values of each driver instead of only the resolved value (see Simulator Run-Time Options on page 17-24).

For register variables, which are commonly assigned values by multiple statements throughout the course of a simulation, it is exceedingly difficult to determine which line of code correlates to a particular value change using standard debug facilities. The Event Origin feature can leverage some of the more advanced capabilities of VirSim to help with these problems. Saving line trace data can eliminate the consideration of drivers that were not executed at the time of the change (see Chapter 17, VCD+ (vpd) File Generation).

13

Building Buses

Use VirSim's Bus Builder function to easily create and customize buses.

User-created buses can contain signals as well as other user-created buses. Created buses then can be expanded, collapsed, and accessed in the Hierarchy Browser in the same way as vectors, memories, or records.

Using Bus Builder

Access the Bus Builder dialog from the Edit pull-down menu in either the Waveform or Register windows.

By dragging signals into the Bus Builder Dialog, the component signals are displayed in the dialog and the appropriate default values are set. You can modify any of the dialog settings and then click Ok to create the bus.

All components of a bus must be signals from the same history file. Buses can not have components that are expressions. You can not edit a part select using the Bus Builder Dialog. You can create a bus that contains just the desired range of bits, but it will not follow the conventions for part-select names. It must be a valid bus name.

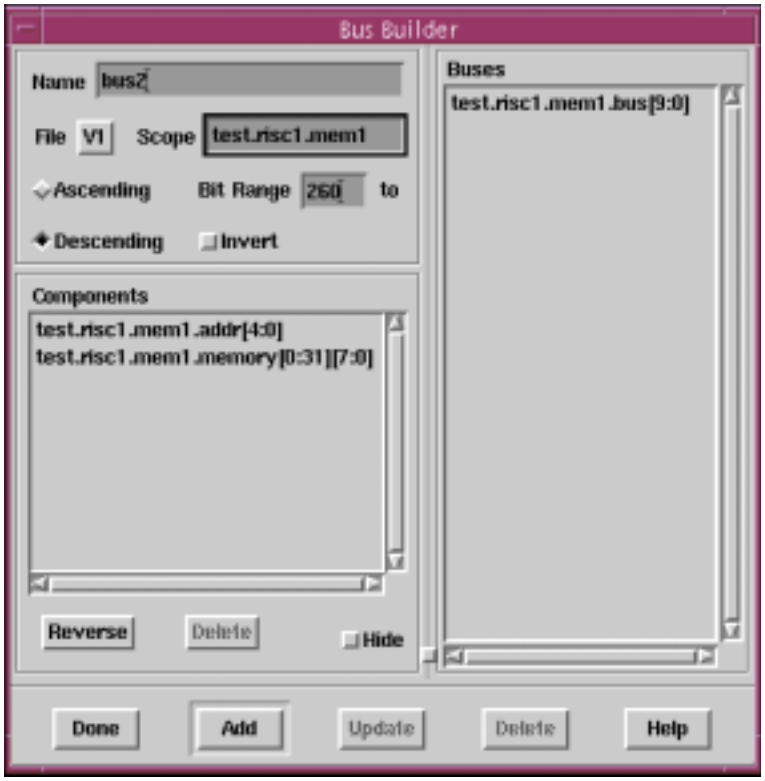
After a bus is created, it can be used as any other signal in the design. By default it will reside in the highest level scope common to its components. Drag its name from the Bus Builder dialog to the Hierarchy Browser Signal Select pane to see its home scope. Drag it to a Waveform Window or Register Window to display its data.

To create a new bus	Drag components from any window into the Component list. Modify the dialog settings as necessary. Click Add to create the bus.
To edit an existing bus	Select the bus from the Buses window. Modify the settings and add and delete components as necessary. Click Update to make the changes to the bus.
To delete a component from a bus	Select a component, then either click the Delete button under the Component pane or press the Delete key on your keyboard.
To delete a bus	Select a bus from the Buses list and click the delete button on the Bus Builder dialog.
To view buses from another open file	Click the file menu and select the file designator for the file you want to view.

Bus Builder Dialog

Figure 13-1, Bus Builder Dialog, shows the major areas of the Bus Builder Dialog.

Figure 13-1 Bus Builder Dialog



Name

Buses can be named any legal name for the language (e.g., Verilog, VHDL). The bit ranges should not be specified as part of the name as they are added automatically (like any other vector). If no name is specified, by default buses are numbered consecutively within each design file (Bus1 through BusN).

File

Specifies the file designator for the source history file. File designators are selected from the pull down menu next to the File field. As you switch files, the Bus Builder dialog lists only the buses associated with the selected file. You can not drag a component from one file into a bus created from another file.

Scope

When a bus is created, it is associated with a scope in the design. The bus will be displayed in the Hierarchy Browser Signal Select pane for its associated scope. The Scope field allows you to modify the default scope. The default scope is determined by the following:

- If all components are from the same scope, that is the default scope.
- If all components have a common ancestor, the scope is that ancestor.
- If there are no common ancestors, the scope will be the root scope.

Ascending/Descending

Specifies whether the bits are numbered in ascending order (0-5) or descending order (5-0).

Bit Range

By default the bit range is 0 to N. You may edit these values.

Invert

Inverts each data bit in the bus. For example, 4’b1010 inverted would become 4’b0101.

Components

Lists the components of the currently selected bus. You can drag signals into the components list from the Waveform Window and Hierarchy Browser.

Reverse

Reverses the order of selected bus subcomponents.

Delete (below Components list)

Deletes selected components.

Hide

Bus components are displayed in the Hierarchy Browser's Signal Select list only as part of that bus. When Hide is off, signals are displayed in the Signal Select window both individually and as components of the bus.

Buses

Lists all defined buses for the selected file.

Done

Closes the Bus Builder Dialog.

Add

Creates a new bus and displays it in the Buses List.

Update

Updates an existing bus.

Delete

Deletes an existing bus. Use the delete key to delete components of a bus.

Help

Opens on-line help for the Bus Builder Dialog.

Using Buses in the VirSim Windows

Logic Browser

The Logic Browser can not show a bus made up of different signals. When a bus is loaded into the Logic Browser, the Connection Dialog allows you to select which component you want to track.

Source Window

When a bus is dropped into a Source Window it is displayed in the bus' scope. If the bus uses the artificial root as the scope, then the following message is displayed:

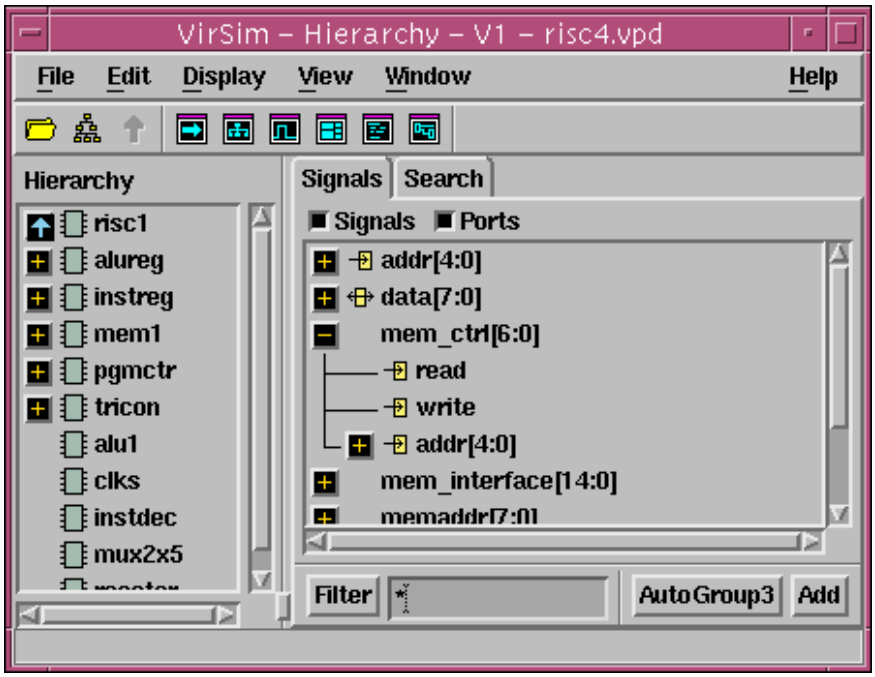
> Bus <busname> does not have a scope.

Hierarchy Browser

When you create a bus it appears in the Hierarchy Browser Signal Select pane when its associated scope is selected. Click on a bus in the Hierarchy Browser Signal pane to expand it. When expanded, a bus shows the components which, if they are not scalars, can be further expanded.

If a bus was created with Hide selected, components that make up that bus are displayed in the Hierarchy Browser only as part of the bus. The individual components are hidden from view in the Hierarchy Browser.

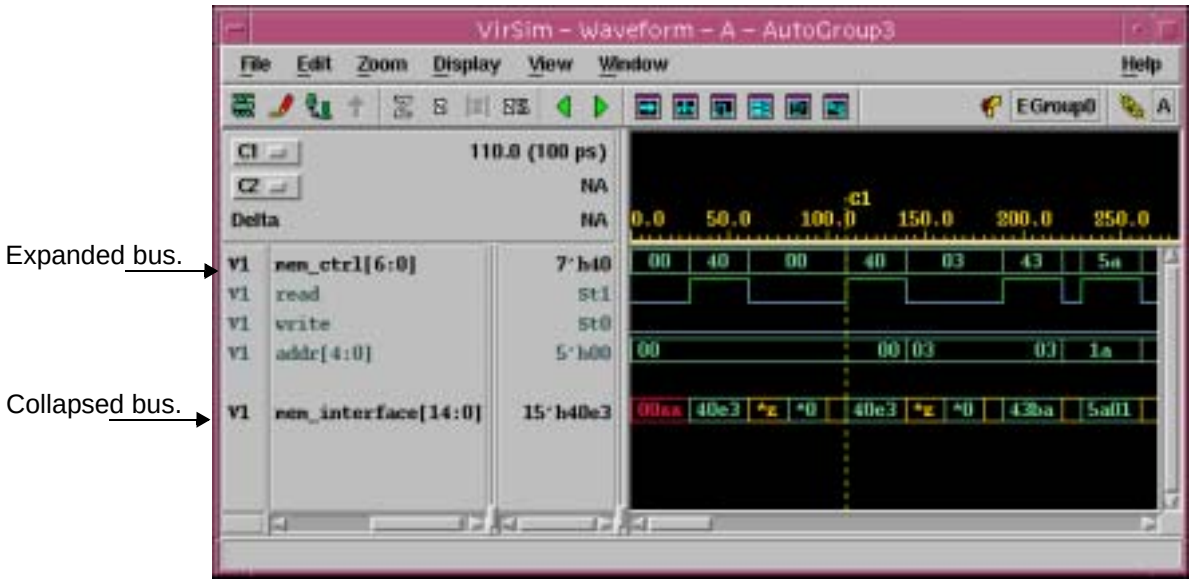
Figure 13-2 Expanded Bus in Hierarchy Browser Signal Select Pane



Waveform Window

Double click on a bus name in the signal name pane to expand it and show its individual components. Double click again to collapse a bus.

Figure 13-3 Buses in the Waveform Window



14

Markers

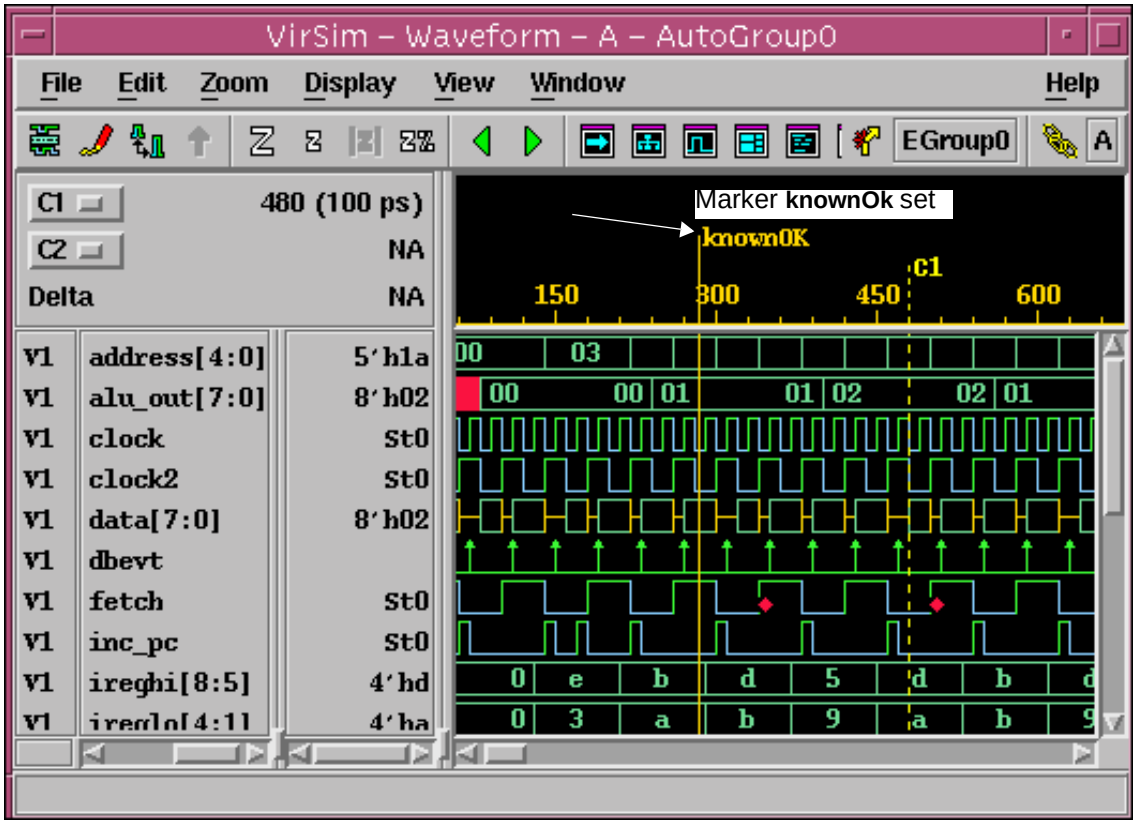
Markers function as user-defined aliases for specific simulation times. They are used to bookmark specific points in simulation time in VirSim windows so that specific simulator times can be easily accessed.

How VirSim Uses Markers

Markers are displayed in the Waveform Window and can be used to set time in the Waveform, Register, and Source Windows. If any of these windows are linked, setting time to a marker in one window changes the time displayed in all linked windows.

In the Waveform Window, a marker displays as a solid vertical line in a Waveform pane with the marker name at the top (see Figure 14-1, Marker in Waveform Window).

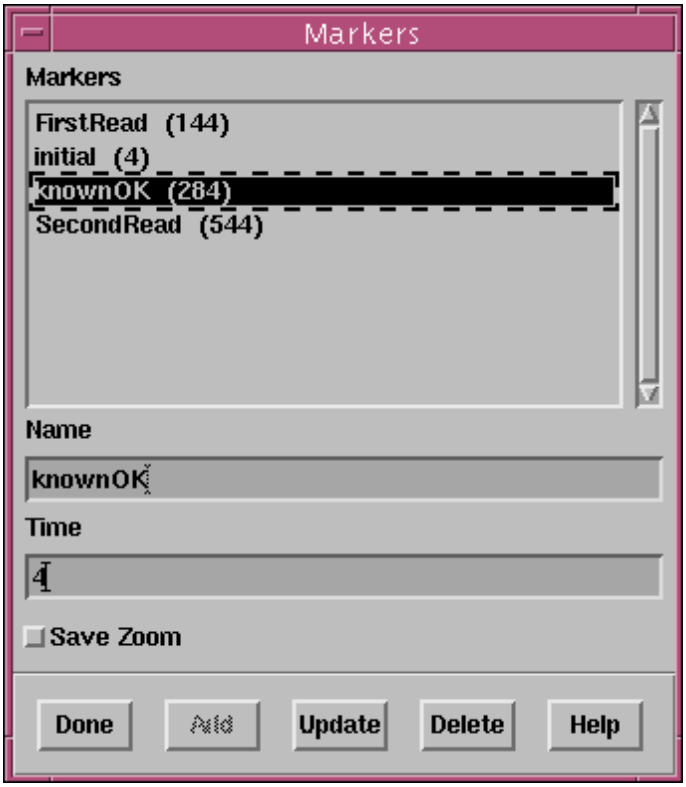
Figure 14-1 Marker in Waveform Window



Markers Dialog

The Markers Dialog is used to create, edit, or delete markers (see Figure 14-2, Markers Dialog).

Figure 14-2 Markers Dialog



The Markers Dialog has a Markers list, a Name text field, a Time text field, a Save Zoom toggle button, and five action buttons.

Markers (List)

Displays the names of all defined markers. When you select a marker name in this area, all fields of the Marker Dialog display values associated with that marker.

Name (Text Field)

Used to enter the name of a new or existing Marker. The name may only contain alpha-numeric characters and underscores and must begin with an alphabetic character.

Time (Text Field)

When the Marker Dialog is opened, the Time field contains the time at C1. If you have multiple Waveform Windows open, the last placement of C1 is displayed. You can use the C1 time or enter a new time.

Save Zoom (Toggle Button)

If the Save Zoom is toggled on, VirSim saves the zoom factor with the marker for the current Waveform Window. Subsequently, when you go to that marker, the zoom factor is restored. This field is sensitive only when the Markers Dialog is opened from the Waveform Window.

Done

Closes the Markers Dialog.

Add

Creates a new marker.

Update

Changes the marker name, time, or zoom setting of an existing marker.

Delete

Deletes an existing marker.

Help

Opens on-line help for the Markers Dialog.

Opening the Marker Dialog

In the menu bar, click left on Edit and choose Markers.

Creating a Marker

To create a marker in the Markers Dialog:

1. Click left in the Name field and enter a Marker name.
2. Click left in the Time field and enter a simulation time or default to current C1 time.
3. If you want to save the zoom factor, click left on the Save Zoom toggle button. A pressed button indicates that the zoom factors will be saved.
4. Click left on the Add button to save the marker.

Deleting a Marker

To delete a marker:

1. Click left on a marker name in the Markers field.
2. Click left on the Delete button to delete the marker.

Editing a Marker

To edit a marker in the Markers Dialog:

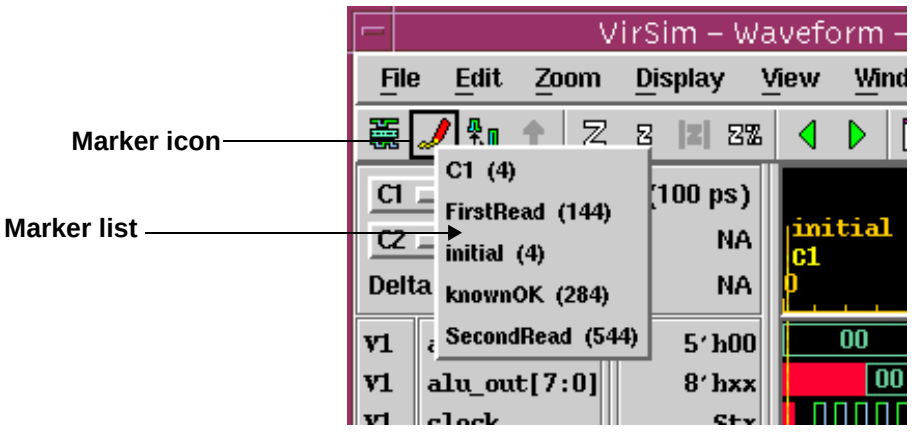
1. Click left in the Name field and enter a marker name.
2. Click left in the Time field and enter a simulation time.
3. To save the zoom factor, click left on the Save Zoom toggle button.
4. Click left on the Update button to update edit changes to the marker.

Setting a Marker

The following procedure explains how to set a marker. You can set a marker in either the Waveform Window, Register Window, or Source Window.

1. Click left on the Marker icon in the toolbar. A list of configured markers appears.

Figure 14-3 Setting a Marker



2. Click left on a marker name in the list. All linked windows display simulation data at the time designated by the marker.

15

Expressions

Expressions are used to create pseudo signals and search for events. Expressions are created with Verilog or VHDL expression syntax and operators. You can search for events that are either level sensitive, edge-triggered, or a combination of both.

Expressions can be used for searches in the Waveform Window and results can be displayed in both the Waveform and Register windows.

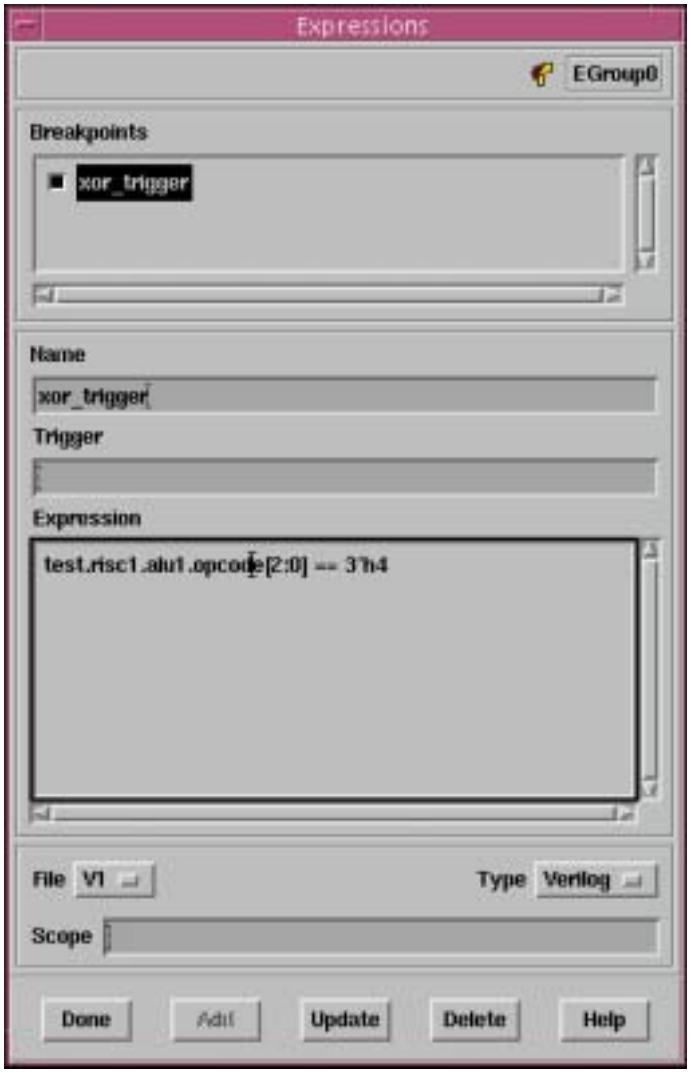
Note:

The VHDL simulator handles only true and false expressions, whereas VirSim and Verilog allow expressions based on signal value. If you are using VHDL or mixed VHDL/Verilog with a Scirocco simulator, be sure that expressions are written for a True/False result. For example, use (signal1 or signal2) = ’1’ rather than (signal1 or signal2).

Expressions Dialog

The Expressions Dialog is used to create or edit an expression. Figure 15-1, Sample Expressions Dialog, shows an example of an expression definition in the Expressions Dialog. The expression must have a name and an expression or trigger statement that defines the search criteria. In the example, the expression searches the opcode signal for a signal value of 3'h4.

Figure 15-1 Sample Expressions Dialog



The Expressions Dialog has a Breakpoints list, a Name field, a reference File designator menu, a reference Scope field, a Trigger field, an Expression field, and five action buttons.

Breakpoints

Displays the names of all expressions that have been entered and indicates if the expression is being used for searching in the currently active breakpoint group.

- Use the Breakpoint icon to select among the defined breakpoint groups. Note that the breakpoint group associated with any open window is not affected by toggling the group displayed in this dialog.
- To display values in all fields associated with the expression, select an expression name in the Breakpoints List. All fields of the dialog display values associated with this expression.
- To enable searching on an expression, click left on the toggle to the left of the expression name.
- To display expressions in the Waveform or Register Window, drag and drop the expression name from this area to the window.

Name

Lists the name of the expression. Use Name to name or rename the expression. Names can include any combination of alphanumeric characters and underscores, but they must not begin with a number.

Use of short names is advised. When the expression completes a search in the Waveform Window, the expression name is displayed above the C1 cursor.

File

Specifies the file designator for the file associated with signal name. File designators are selected from the pull-down menu next to the File field. Not all file types support expressions. The File field displays only those open files that support expressions. If the Trigger or Expression field contains the file designator, the File field is ignored (see Scope below).

Scope

Specifies a scope from which signal names in the Trigger and Expression fields are referenced. If the Trigger and Expression fields include the file designator and full hierarchy names of the scope, the Scope field is ignored.

For example, the following expression does not require a file designator and scope because it includes the file designator and full hierarchy name:

Verilog:

```
($V1$test.risc1.alu_out[7:0])===($V1$test.risc1.accum[7:0])
```

VHDL:

```
$V1$test/risc1/alu_out(7 Downto 0)===
                          $V1$test/risc1/accum(7 Downto 0)
```

The following expression requires a file designator and scope because it does not include the file designator and full hierarchy name:

Verilog:

```
(alu_out[7:0])===(accum[7:0])
```

VHDL

```
alu_out(7 Downto 0)===accum(7 Downto 0)
```

When the expression does not include the full hierarchy name, VirSim looks for all signals in the referenced scope. If you drag in a signal from another window, VirSim automatically adds the file designator and full hierarchy name to the expression.

Verilog Trigger

The trigger is edge-sensitive. The trigger can use two Verilog edge operators: posedge and negedge or, if the trigger uses just the signal name, any transition of that signal satisfies the expression. If it uses `posedge`, then only the rising transitions (x->1, 0->1, z->1) satisfy the expression. If it uses `negedge`, then only the falling transitions (x->0, 1->0, 1->z) satisfy the expression.

VHDL Trigger

The trigger is edge-sensitive. The trigger can use VHDL signal attribute `S'event`. For example, `clk'event` and `clk==1`--rising edge clk, and `clk'event` and `clk==0`--following edge clk. If the trigger uses just the signal name, any transition of the signal satisfies the expression.

Expression

The expression is level-sensitive. To enter an expression, type the expression, drag and drop in signals, and insert operators. For signals that you drag and drop, the file designator and full hierarchy name are automatically inserted in the expression.

Done

Closes the Expressions Dialog.

Add

Creates a new expression and displays it in the Breakpoints List.

Update

Updates edit changes to an existing expression.

Delete

Deletes an existing expression.

Help

Opens on-line help for the Expressions Dialog.

Entering Signal Names

To mix signals from different open files, the file designator must be prefixed to the signal name. For example, for Verilog a file designator of V2 and a signal name of `test.risc1.reset` would be entered as:

```
$V2$test.risc1.reset
```

in the Expression or Trigger fields.

Other signal naming considerations include:

- Signals dragged and dropped from other windows automatically include the file designator in the hierarchy name.

- Signals that have the same file designator as the File option do not need the designator added as a prefix to the hierarchy name.
- Signals from different file types (for example, VCD+ and EPIC) cannot be mixed in the same expression.

Opening the Expressions Dialog

In the menu bar, click left on the Edit menu and choose Expressions.

Creating Expressions

To create a new expression:

1. Click left in the Name field and enter an expression name.
2. Choose a file designator from the File pull-down menu. (This field is optional if the expression includes the full hierarchy name with designators.)
3. Click left in the Scope field and enter the full hierarchy name of the scope you are using for signal selection. For example:

   Verilog

   ```
   $V1$test.risc1.alu1 or $test.risc1.alu1
   ```

   VHDL

   ```
   $V1$test/risc1/alu1 or $test/risc1/alu1
   ```

   Instead of typing, you can drag and drop a scope from the Hierarchy Window into the Scope field. (This field is optional if the expression includes the full hierarchy name.)

4. Click left in the Trigger field and enter an edge-triggered expression. (If the Expression field is used, this field is optional.)
5. Click left in the Expression field and enter a level sensitive expression. (This field is optional if the Trigger field is used.)

   For more details on how these expressions behave, see the level sensitive and edge-triggered search examples later in this chapter.

6. To assign expressions to a breakpoint group, click left on the breakpoint group icon (in the upper-right corner) to open a menu of available breakpoint groups and choose the desired group.
7. In the Breakpoints list, enable expressions for searching. Click left on the toggle button for each expression that you want to include in the expression group. When the expression is enabled, the toggle button is back lit.
8. Click left on the Add button. To add additional expressions, repeat steps 2 through 7.

Updating Expressions

To update edit changes to expressions:

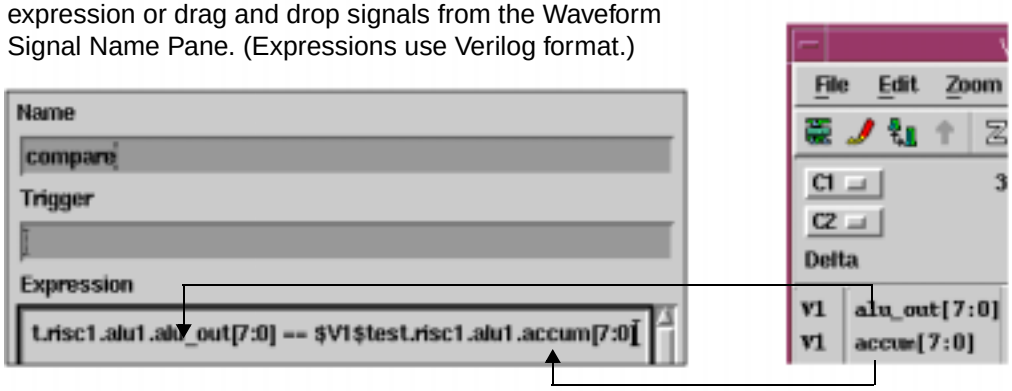
1. Click left on the expression name in the Breakpoints list. All the fields are filled in for editing.
2. Edit values in the Scope, File, Trigger, and Expression fields as required.
3. After editing the expression, click left on the Update button.

Displaying Expressions

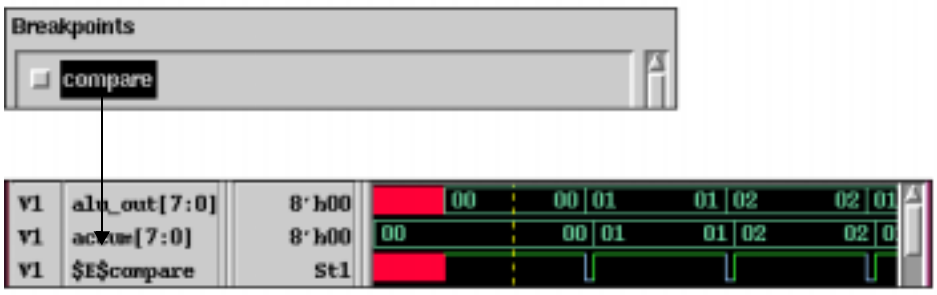
To display the signal generated by an expression, drag the expression name from the Expressions Dialog to the Signal Name Pane of the Waveform Window or to the Register Window. A E prefix is added to the expression name and a T prefix is added to the trigger name. The expression signal indicates level sensitive changes in the waveform. Figure 15-2, Displaying Expressions Example, shows how expressions are displayed.

Figure 15-2 Displaying Expressions Example

1. Create expression in Expressions Dialog. Either type the expression or drag and drop signals from the Waveform Signal Name Pane. (Expressions use Verilog format.)



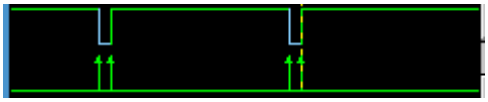
2. Drag and drop the expression name to the Waveform Name Pane. The expression signal appears in the Waveform Pane.

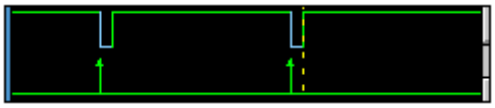


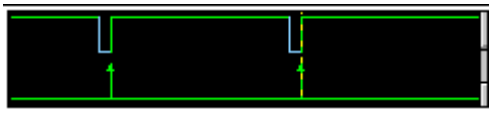
Trigger Types

The trigger expression can specify the negative edge, positive edge, or both edges of a signal to trigger on. To specify the edge to trigger on, the trigger expression should include the appropriate event-control statement (see Table 15-1, Trigger Expressions).

Table 15-1 Trigger Expressions

Trigger Expression	Signal Edge
`clkt`	Trigger on both negative and positive edge. 

Verilog:
`negedge clkt`
 	
VHDL:
`clk'event and clk='0'`
	Trigger on negative edge. 

Verilog:
`posedge clkt`
 	
VHDL:
`clk'event and clk='1'`
	Trigger on positive edge. 

Searching with Expressions

There are two search arrow icons in the Waveform Window toolbar that allow you to search forward or backward from the C1 cursor. Their search parameters are set by activating the desired expressions or breakpoint groups in the Expressions Dialog.

Multiple expressions can be activated concurrently in any breakpoint group, but only one breakpoint group may be active per window link. The first expression that finds an event stops the search, and the expression name is displayed above the C1 cursor name. If multiple expressions find events at the same simulation time, the name of the first expression in the list is displayed with an asterisk. Pressing the Search icon in the same direction then causes C1 to display the name of the next expression found and so on. Also see Searching a Signal (Vector or Scalar) for a Specified Value on page 4-17.

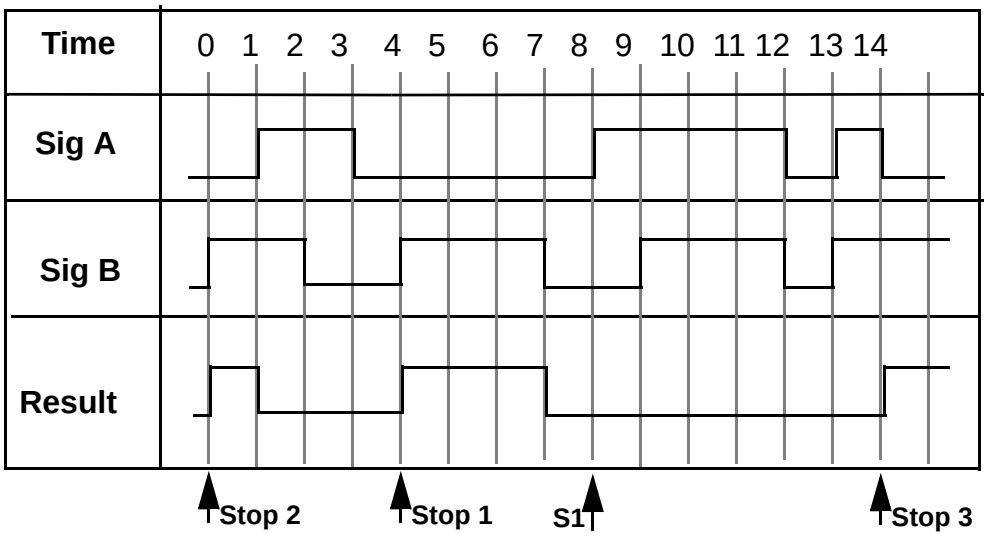
The following types of searches are covered in this section:

- Level Sensitive Searches
- Edge-triggered Searches

Level Sensitive Searches

Level sensitive expressions search for when events defined in the expression first become true. These expressions do not have a trigger.

Level Sensitive Searching Example:



The search criteria is A low and B high. The expression searches backwards from S1 at time 8.

VirSim stops at time 4 (Stop 1) the first time where the criteria became true. If you push the search backwards arrow again from time 4, VirSim stops at time 0 (Stop 2). The level sensitive search finds any point where A is 0 and B is 1.

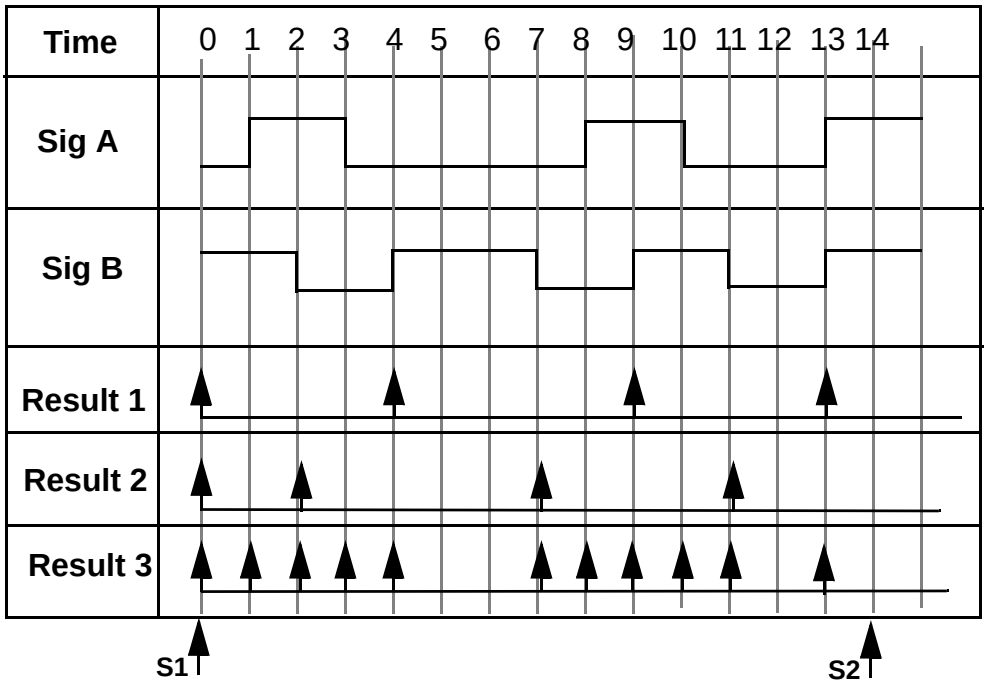
If you search forward from S1, VirSim stops at time 14 (Stop 3) where the search criteria first became true. To view the results of the expression in the Waveform Pane, drag the expression name into the Signal Name Pane of the Waveform Window.

Edge-triggered Searches

Edge-triggered expressions search for the specified edge, then check the level sensitive condition designated in the expression.

The signal generated for trigger expressions is a vertical arrow for each time the trigger condition occurs.

Edge-Triggered Searching Example:



- Searching forward from S1 for B positive edge produces Result 1.
- Searching backward from S2 for B negative edge produces Result 2.
- Searching forward from S1 for A or B produces Result 3.

16

VirSim Setup

You can individually customize VirSim by defining settings for certain VirSim features and screen colors in the appropriate X Resource file or Windows NT Registry.

X Resources enables the user to run VirSim for UNIX with Motif™. VirSim settings are defined in the VirSim specific application defaults file named VirSim.

Caution: Before You Start

Whenever you change the X Resource file, always make a backup copy. For NT systems, use the Export feature of the Registry editor to back up Registry settings prior to editing. Use the existing VirSim NT Registry or X Resource file as the basis for changes to settings to assure that the formatting is correct. Changes are not checked for validity and may have adverse affects. Changes mistakenly entered in non-VirSim files and variables can affect the performance of your system in unpredictable ways. Contact your system administrator for information about changing your NT Registry or X Resource files.

Defaults have been carefully chosen for good contrast and aesthetic appeal. In the interests of quality product support, it is wise to limit the number of changes.

VirSim Platforms

VirSim is designed to run on the following operating system platforms. Each platform has a specific mechanism for defining settings for application behavior.

These platforms include:

- OSF/Motif for UNIX
- Windows NT

X Resources

X Resources enables the user to run VirSim for UNIX with Motif™. VirSim settings are defined in the VirSim specific application defaults file named VirSim. For information on setup for X Resources, see the section Settings for X Resources on page 16-15.

To modify selected settings on a per-user basis, add the selected variables from the `$VIRSIMHOME/appfiles/VirSim` file into your `~/.Xdefaults` file and modify the settings. Then execute `xrdb -m ~/.Xdefaults` to merge the new variables into your X environment. Your new settings should take effect the next time you start a VirSim session.

For example, putting the following line in your `.Xdefaults` file:

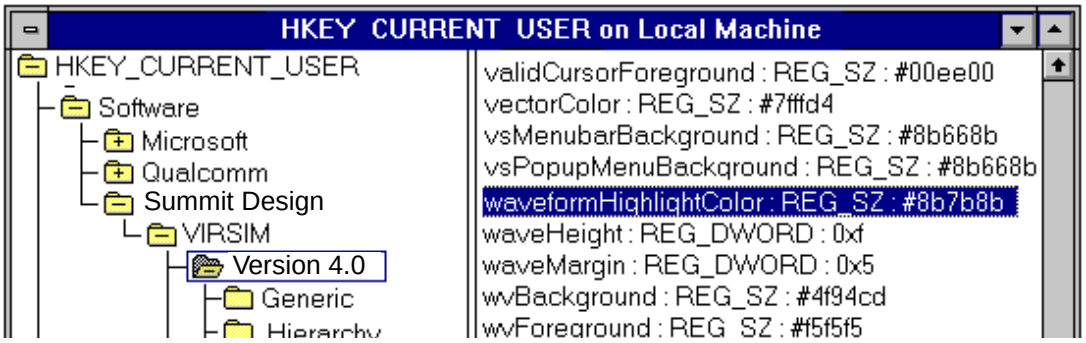
```
enableWaveformHighlight: True
```

will cause the actual waveforms of selected signals in the Waveform Window to be highlighted in a grey color.

Windows NT Registry

In the Windows NT™ version, VirSim settings are defined in the Registry (regedit). See Figure 16-1, VirSim Setup in Registry. To access regedit, choose Run from the Start menu and type regedit in the Open field.

Figure 16-1 VirSim Setup in Registry



Common Settings

This section lists VirSim settings that you may wish to change. It is organized in the following sections:

- Mouse Button Settings (Windows NT only)
- Numeric and Vector Separator Settings
- Warning if Configuration is Modified
- Waveform Window Settings
- Logic Browser Settings (Verilog Only)
- Register Window Settings
- Interactive Window Settings

In Windows NT, you must run VirSim once before you can change any settings. The first run sets the default values for all entries.

Note:

In X Resources, the variable name is preceded by the prefix VirSim. For example:

```
VirSim.lbModCellColor: Khaki4
VirSim*lbMainFormFM*lbGraphicsDA.background: Black
```

Different platforms use different values for variables. Only X Resources can use color names as values. MainWin and Windows NT require a hex value to identify the color. For example:

In X Resources:

```
rwSignalValueForegroundColor: WhiteSmoke
```

In Windows NT

```
rwSignalValueForegroundColor: #f5f5f5
```

WindowsNT uses the RGB format *#RRGGBB*.

Mouse Button Setting

(Windows NT Only) VirSim is designed for use with either a two-button or three-button mouse. The MouseButtonMode variable is used to set the mouse type.

MouseButtonMode: *n*
Where n is:

2 for a two-button mouse
3 for a three-button mouse
A for Auto. Auto automatically detects the mouse type

Note:

Auto mode is only available on Windows NT systems. However, Windows NT may not be able to accurately detect the mouse type 100% of the time because of known NT configuration problems.

Numeric and Vector Separator Settings

The `vectorSeparator` variable sets the grouping separator for binary, octal, decimal, or hexadecimal vector values. Legal separators are ",", "_", and ".". With the separator set to "_" and a spacing of 2, a 32-bit value of all-ones is displayed as hexadecimal ff_ff_ff_ff.

```
vectorSeparator: _
```

The spacing variables listed below set the number of digits between each vector separator for each radix. A value of zero turns off the separator for that radix.

```
binarySpacing: 8
octalSpacing: 4
decimalSpacing: 3
hexSpacing: 2
```

The numericSeparator variable sets the decimal separator for all time values. Legal separators are ",", "_", or "none". With the separator set to a comma, a time value of 150000.50 is displayed as 150,000.50.

```
numericSeparator: ,
```

The timeSpacing variable sets the number of digits between each numeric separator.

```
timeSpacing: 3
```

Warning if Configuration Is Modified

The warnIfConfigurationModified variable controls the display of the warning dialog: `Current configuration has changed.` VirSim can be configured to display the dialog when you exit without saving the configuration. When True, the dialog displays when the configuration has changed. The default value is `False`.

```
warnIfConfigurationModified: False
```

Signals Loaded/Registered with Simulator

The isDeferredLoadData variable controls the registering of signals in interactive mode and the loading of signals from the vpd file in post processing mode.

When True, data is registered or loaded only for those signals that are displayed in the currently active Waveform signal group or Register Window view. This setting reduces memory usage.

When False, all signals for all defined Register Window views and Waveform Window signal groups are registered or loaded, regardless of whether they are currently being displayed.

The default value is `False`.

```
isDeferredLoadData: False
```

User Defined Unique Event Colors

(Verilog only) Table 16-1, User Define Unique Event Colors, shows settings for colors that signify the severity of unique events.

Table 16-1 User Define Unique Event Colors

Item	Variable
Error Unique Event	`dsiColor67: red`
Warning Unique Event	`dsiColor68: yellow`
Information Unique Event	`dsiColor69: green`

Waveform Window Settings

Table 16-2 and Table 16-3 show settings for the Waveform Window. Colors for `dsiColor01` to `dsiColor25` appear in the Waveform Style Dialog.

Table 16-2 Settings for Waveform Window (Verilog and EPIC)

Item	Variable
Unset -- no value set for signal (EPIC)	`dsiColor03: gray`
Z signal Value for scalar and vector encoding types (Verilog and EPIC)	`dsiColor03: gold`
0 signal value for a scalar encoding type (Verilog and EPIC)	`dsiColor03: SkyBlue1`
Data signal value for a vector encoding type (Verilog); U for unset or Z for logic Z (EPIC)	`dsiColor03: aquamarine`
Any signal value for a real encoding type (Verilog)	`dsiColor03: cyan`
Any signal value 1 for a scalar encoding type (Verilog)	`dsiColor03: green`
X signal value for an scalar encoding type (Verilog); **All unset** (EPIC)	`dsiColor03: red`
Enable background color of displayed waveforms	`enableWaveformHighlight: True`

Item	Variable
Waveform background color This color is used as background for signals or for highlighting signals if one of these modes is enabled.	`waveformHighlightColor: Thistle4`
Enable waveform zoom centered around C1. When True, the Waveform Window zooms in centered on cursor C1. When False, the Waveform Window zooms in on the center of the screen	`enableWaveformHZoomInOnC1: False`
Enable waveform highlight line. When True, waveforms for signals that are selected are highlighted as well as the signal name. When False, waveforms are not highlighted.	`enableWaveformHighlightLine: False`
Delta Cycle Width The width of a delta cycle is determined by a variable in the VirSim resource file. It sets the number of pixels to draw between each delta cycle. The Default is eight, which corresponds to eight pixels. The minimum is one pixel per delta cycle.	`VirSim*deltaCycleWidth: 8`

Table 16-3 Settings for Waveform Window (VHDL)

Data Type	Value	Variable
STD_Logic	'U'	`dsiColor03: Gray`
	'X'	`dsiColor23: Red`
	'0'	`dsiColor13: Blue`
	'1'	`dsiColor18: Green`
	'Z'	`dsiColor06: Gold`
	'W'	`dsiColor03: Red`
	'L'	`dsiColor13: Blue`
	'H'	`dsiColor18: Green`
	'-'	`dsiColor24: Brown`
Bit	'0'	`dsiColor13: Blue`
	'1'	`dsiColor18: Green`
BOOLEAN	FALSE	`dsiColor18: Green`
	TRUE	`dsiColor18: Green`
Other scalar types	Any value	`dsiColor18: Green`
All vector types	Any value	`dsiColor18: Green`

Source Window Settings

Table 16-4 shows settings for the Source Window.

Table 16-4 Settings for Source Window (Verilog and VHDL)

Item	Variable
Line and block comments	`dsiColor26: Khaki1`
Keywords	`dsiColor27: LightPink1`
Source Data	`dsiColor28: SeaGreen1`
Character Strings	`dsiColor28: SeaGreen1`
Identifiers	`dsiColor30: SeaGreen1`
System tasks	`dsiColor32: LightPink1`
Compiler directives	`dsiColor33: LightPink1`
Truncation selection	`VirSim*truncateOnLeft: 0`

Hierarchy Window Settings

Table 16-5, and Table 16-6, show settings for the Hierarchy Window.

Table 16-5 Settings for Hierarchy Window (Verilog)

Item	Variable
Foreground text	`dsiColor37: Black`
Module scope	`dsiColor38: Gray`
Task scope	`dsiColor39: SkyBlue2`
Function scope	`dsiColor40: LightPink1`
Named begin	`dsiColor33: LightPink1`
Named fork	`dsiColor33: LightPink1`
EPIC foreground	`dsiColor33: LightPink1`
EPIC background	`dsiColor33: LightPink1`
Root scope	`dsiColor33: LightPink1`

Table 16-6 Settings for Hierarchy Window (VHDL)

Scope	Color	Mapping
Root or Parent Level	PaleVioletRed1	`hbTopBackgroundColor`
Entity/Architecture	Gray	`dsiColor38`
Block	SkyBlue2	`dsiColor39`
Procedure	PaleGreen2	`dsiColor40`
Function	PaleGreen2	`dsiColor40`
Package	Gold2	`dsiColor41`
Process	Papayawhip	`dsiColor42`
Foreground text	Black	`dsiColor37`

Logic Browser Settings

Table 16-7 shows settings for the Logic Browser. Windows NT format uses hex values to represent colors.

Table 16-7 Settings for the Logic Browser

Item	Variable
Logic Browser text font	`ibTextFond: fixed`
Maximum number of characters in displayed values Increasing this value increases the size of the layout	`lbwValMaxChars: 10`
When True, shows **value of base spec** (for example, .8'h in 8'h7f)	`lbwValShowBaseSpec: True`
Truncation selection True truncates values on lsb end	`lbwValTruncMsb: False`
Automatically load value changes When True, automatically loads value changes for all ports in Logic Browser	`lbwAutoLoadValueChanges: True`

Register Window Settings

Table 16-8 shows settings for the Register Window:

Table 16-8 Settings for Register Window

Item	Variable
Color of **label**	`rwLabelForegroundColor: Khaki`
Color of **label background**	`rwLabelBackgroundColor: Khaki`
Color of **signal**	`rwSignalValueForegroundColor: WhiteSmoke`
Color of **signal foreground**	`rwSignalValueBackrgoundColor: Black`
Color of **box**	`rwBoxForegroundColor: Green2`
Color of **line**	`rwLineForegroundColor: Green2`
Color that indicates a **signal change value**	`rwLSignalValueChangedColor: HotPink`

Interactive Window Settings

Table 16-9, Settings for Interactive Window, shows settings used to change the default behavior of the Invocation Dialog and simulation start-up in interactive mode. These resources are shown below with their default values.

There are specific defaults for each supported simulator (for example, vxlDefaultSimulator, and vcsDefaultSimulator). There is a specific iwbuttons file for each type of simulator supported (for example, `iwbuttons.vxl`).

Table 16-9 Settings for Interactive Window

Item	Variable
Name of the default simulator The name can be any VirSim supported simulator.	`defaultSimulatorType: VXL`
Name of the Verilog XL simulator The name can be a file name or an absolute path name.	`vxlDefulatSimulator: <path_to_verilog>/verilog`
Default arguments for the simulator Arguments can be edited in the Invocation Dialog	`vxlDefaultSimulatorArgs:`
Step Time field value	`defaultStepTime: 20`
X Resource save lines The value limits the number of lines saved in the History window. The minimum value is 10, the maximum value is limited by available memory.	`saveLines: 1000`
User-defined Button files Simulator-specific files containing the definitions to be used for interactive buttons	`vcsUserButtonFile: -peter/ .iwbuttons.vcs`

Table 16-9 Settings for Interactive Window (Continued)

Item	Variable
Force time synchronization for all windows Default is true. Changing the setting to false allows you to move backwards in time in all windows except the Source Window without unlinking from SIM	`forceSimLinkSync: true`

Settings for X Resources

By default, VirSim automatically uses the X Resource file installed with VirSim. To customize this file for individual users, you must replace the VirSim script with your own version and customize it as desired.

Tool Tips

Tool tips and the tool label appear when you position the cursor over a button in any of the VirSim toolbars. Tool tips are automatically enabled on Windows NT. In Motif, tool tips must be enabled as described below.

To enable this feature, set the following entry to `True`.

```
VirSim*enableToolTips: True
```

Some window managers require additional configuration for correct tool tip operation. The setup for two common window managers is described below:

MWM

No additional setup required.

OLWM

Set the following X resource:

```
olwm.MinimalDecor: ToolTipWindow
```

You may add ToolTips to an existing minimal decorations setting. For example:

```
olwm.MinimalDecor: xclock ToolTipWindow
```

X Color System

The following briefly describes the X color system and how VirSim uses this system:

- Color Model
- Switching Color Maps

For more details on this complex subject, refer to the *Xlib Programming Manual*, listed in the Preface.

Color Model

The two structures are used by X to define the color model used on a system: visual and the colormap.

visual

The visual determines if the system can use color or shades of gray, and if the colors are fixed (read-only) or changeable (read-write). VirSim uses the default visual defined by the X server.

colormap

The colormap designates the colors available on a display. In most color systems, the colormap contains up to 256 color cells, each of which may contain a unique color. Each application you run typically uses up some of the available color cells. When all color cells have been used up, the VirSim application creates a custom colormap. The custom colormap created by VirSim minimizes the technicolor effect for applications that do not use the current colormap. Because most systems support only a single colormap, you may need to switch color maps when moving between applications.

Switching Color Maps

The method used to switch between color maps depends on which window manager is used. The following is a summary of the mwm and olwm window managers.

Switch Color maps with mwm

The mwm supports several methods of switching between different color maps. We recommend setting the mwm resource as follows:

```
Mwm*colormapFocusPolicy: explicit
```

This option requires that the user explicitly switch between color maps. Other resource options include the keyboard and pointer. These options typically result in unacceptable screen flashing and color changes.

Once you have set the colormapFocusPolicy (and restarted mwm), you can change color maps using the mwm Window Menu Button (upper left corner of the border). The menu that pops up should have a color option that initiates installation of the color map. If your mwm popup does not contain color as an option, you can add it by editing .mwmrc and adding the following menu option:

```
"Color" f.focus_color
```

If it becomes necessary for VirSim to create a custom color map, the user must switch to the VirSim color map. To make this switch, set the colormapFocusPolicy, as previously described.

Switching Color maps with olwm

Refer to the olwm man page for more details on color maps. There are several options for configuring olwm. In brief, <Ctrl> L2 in the title bar locks the color map.

17

VCD+ (vpd) File Generation

VCD+ files, also referred to as vpd files, are binary files containing simulation history data. They are created from $vcdplus system tasks in verilog simulation and by default are named vcdplus.vpd. System tasks include the vcdplus name in the tasks, for example, $vcdpluson, $vcdplusoff, $vcdplusfilename, etc. These system tasks are entered in Verilog code or at the Interactive Window command prompt. Refer to System Tasks and Functions. There are also simulator run time options that control how VCD+ files are generated. Run time options include +vpd in their names, for example, +vpdbufsize, +vpdignore, etc. Refer to Simulator Run-Time Options.

When you include the -I, -RI, or -PP switches at compile time, VCS automatically links to VirSim's PLI to convert simulator output into the VCD+ file format. Refer to Compile Options for Creating a VPD File from VCS.

For Verilog simulators other than VCS, the correct VCD+ interface must be linked to the simulator in order to produce VCD+ files. This linkage is done as part of VirSim installation and new simulator installation. Refer to the VirSim Installation Notes (separate from this manual).

The Scirocco simulator has a built in interface to create VCD+ files. Refer to the Scirocco Reference Manual for details.

Advantages of VCD+

VCD+ offers significant advantages over the standard VCD ASCII format.

- VCD+ provides a compressed binary format that dramatically reduces file size as compared to VCD and other proprietary file formats.
- The VCD+ compressed binary format dramatically reduces signal load time.
- VCD+ allows data collection for signals or scopes to be turned on and off during a simulation run, thus, dramatically improving simulation run time and file size.
- VCD+ can also save Verilog source-statement execution data. This allows instant replay of your Verilog source execution in the VirSim Source Window.

VCD+ has command-line options that affect performance and file sizes. These options, presented in subsequent sections, allow the user to run VCD+ in the most effective manner. To optimize simulator performance and VCD+ file size, consider the size of the design, the RAM memory capacity of the user workstation, swap space, disk storage limits, and the methodology used in the project.

Note:

For simulators other than VCS, the VirSim PLI must be linked to your simulator executable. Refer to the Installation Notes distributed with the VirSim software.

System Tasks and Functions

VCD+ system tasks are used to capture and save value change data in a binary format so that the data can be viewed in the Waveform Window, Register Window, Source Window, and Logic Browser. The following VCD+ system tasks may be included in source files or entered at the simulator interactive prompt:

System Tasks to Generate a VCD+ File

Note:

The `$vcdpluson` and `$vcdplusoff` system tasks accept the same arguments as the Verilog `$dumpvars` system task. Unlike standard VCD, this lets you turn recording on or off for variables during the same simulation.

$vcdpluson

The `$vcdpluson` task begins recording signal value changes of the specified scope(s) or signals to the VCD+ history file.

Syntax:

```
$vcdpluson (level,scope*,signal*);
```

The descriptions of the task arguments is as follows:

level
: Specifies the number of hierarchy scope levels to desend to record signal value changes (a zero value records all scope instances to the end of the hierarchy; default is all).

scope
: Specifies the name of the scope in which to record signal value changes (default is all).

signal
: Specifies the name of the signal in which to record signal value changes (default is all).

Note: * indicates argument can have a list of more than one value (for scopes or signals).

Example 1: Record all signal value changes.

```
$vcdpluson;
```

Example 2: Record signal value changes for scope `test.risc1.alureg` and all levels below.

```
$vcdpluson(test.risc1.alureg);
```

Example 3: Record two levels of signal value changes: Scope (`test`) and one level below.

```
$vcdpluson(2, test);
```

$vcdplusoff

The `$vcdplusoff` task stops recording the signal value changes for specified scope(s) or signal(s).

Syntax:

`$vcdplusoff (`*`level`*`,`*`scope*`*`,`*`signal*`*`);`

Example 1: Turn recording off.

```
$vcdplusoff();
```

Example 2: Stop recording signal value changes for scope `test.risc1.alu1`.

```
$vcdplusoff(test.risc1.alu1);
```

Example 3: Stop recording signal value changes for `test.risc1.alu1` and `test.risc1.instreg.d1`.

```
$vcdplusoff(test.risc1.alu1, test.risc1.instreg.d1);
```

Example 4: Stop recording signal value changes for scope `test.risc1.alu1` and 39 levels below. In this example, 40 is a number large enough to ensure all lower levels are turned off.

```
$vcdplusoff(40, test.risc1.alu1);
```

Note:

The `$vcdpluson/off` commands increment/decrement an internal counter for each signal to be recorded. If multiple `$vcdpluson` commands cause a given signal to be saved, the signal will continue to be saved until an equivalent number of `$vcdplusoff` commands apply to the signal. This approach allows multiple users to instrument the Verilog sources to meet their individual recording requirements for the same simulation run.

$vcdplusflush

The `$vcdplusflush` task flushes to the VCD+ data file any value changes that have been reported by the simulator but have not yet been written to the VCD+ data file.

Syntax:

```
$vcdplusflush;
```

$vcdplusautoflushon

When simulation stops, the `$vcdplusautoflushon` task automatically flushes to the VCD+ data file any value changes that have been reported by the simulator but have not yet been written to the VCD+ data file.

Syntax:

```
$vcdplusautoflushon;
```

$vcdplusautoflushoff

The `$vcdplusautoflushoff` task turns off the automatic flush (enabled by the `$vcdplusautoflushon` task).

Syntax:

```
$vcdplusautoflushoff;
```

$vcdplusfile

The `$vcdplusfile` specifies a VCD+ file name. If not specified, `VCD.vpd` is default for VHDL and `vcdplus.vpd` is default for Verilog.

Syntax:

```
$vcdplusfile ("filename");
```

$vcdplusclose

The `$vcdplusclose` terminates all tracing, flushes data to file, closes the current VCD+ file, and resets all default settings.

Syntax:

```
$vcdplusclose;
```

System Tasks and Functions for MDAs

This section describes system tasks and functions that provide visibility into multi-dimensional arrays (MDAs).

There are two ways to view MDA data:

- The first method, which uses the `$vcdplusmemon` and `$vcdplusmemoff` system tasks, records data each time an MDA has a data change.
- The second method, which uses the `$vcdplusmemorydump` system task, stores data only when the task is called.

Syntax for Specifying MDAs

The following syntax is required for specifying MDAs using the `$vcdplusmemon`, `$vcdplusmemoff`, and `$vcdplusmemorydump` system tasks:

```
system_task( Mda [, dim1Lsb [, dim1Rsb [, dim2Lsb [, dim2Rsb
[, ... dimNLsb [,  dimNRsb]]]]]] );
```

The description of the syntax is as follows:

system_task
: Name of the system task (required): `$vcdplusmemon`, `$vcdplusmemoff`, or `$vcdplusmemorydump.`

Mda
: This argument specifies the name of the MDA to be recorded. It must not be a part select. If no other arguments are given, then all elements of the MDA are recorded to the VPD file.

dim1Lsb
: This is an optional argument that specifies the name of the variable that contains the left bound of the first dimension. If no other arguments are given, then all elements under this single index of this dimension are recorded.

dim1Rsb
: This is an optional argument that specifies the name of variable that contains the right bound of the first dimension.

  Note: The `dim1Lsb` and `dim1Rsb` arguments specify the range of the first dimension to be recorded. If no other arguments are given, then all elements under this range of addresses within the first dimension are recorded.

dim2Lsb
: This is an optional argument with the same functionality as `dim1Lsb`, but refers to the second dimension.

dim2Rsb
: This is an optional argument with the same functionality as `dim1Rsb`, but refers to the second dimension.

dimNLsb
: This is an optional argument that specifies the left bound of the Nth dimension.

dimNRsb
: This is an optional argument that specifies the right bound of the Nth dimension.

Note that MDA system tasks can take 0 or more arguments, with the following caveots:

- No arguments: The whole design will be traversed and all memories and MDAs will be recorded.

  Note that this process may cause significant memory usage, and simulator drag.

- One argument: If the object is a scope instance, all memories/ MDAs contained in that scope instance and its children will be recorded. If the object is a memory/MDA, that object will be recorded.

Using $vcdplusmemon and $vcdplusmemoff

You can use the `$vcdplusmemon` and `$vcdplusmemoff` tasks to turn on and off, respectively, the recording of changes within memories or MDAs in a design. By using these tasks in VCS, you are able to view changes of memories and MDAs in VirSim windows.

Running VCS

In order for VCS 7.0 to provide MDA data using the `$vcdplusmemon` and `$vcdplusmemoff` tasks, the VCS `+memcbk` and the `+v2k` switches are required.

VCS command line example: `vcs -R -I mda.v +memcbk +v2k`

MDA declaration example: `reg [1:0] mem [3:0] [6:4];`

In order for VCS 7.0 to provide memory data, the VCS `+memcbk` switch is required.

VCS command line example: `vcs -R -I mda.v +memcbk`

Memory declaration example: `reg [1:0] mem [3:0];`

Examples

This section provides examples and graphical representations of various MDA and memory declarations using the `$vcdplusmemon` and `$vcdplusmemoff` tasks.

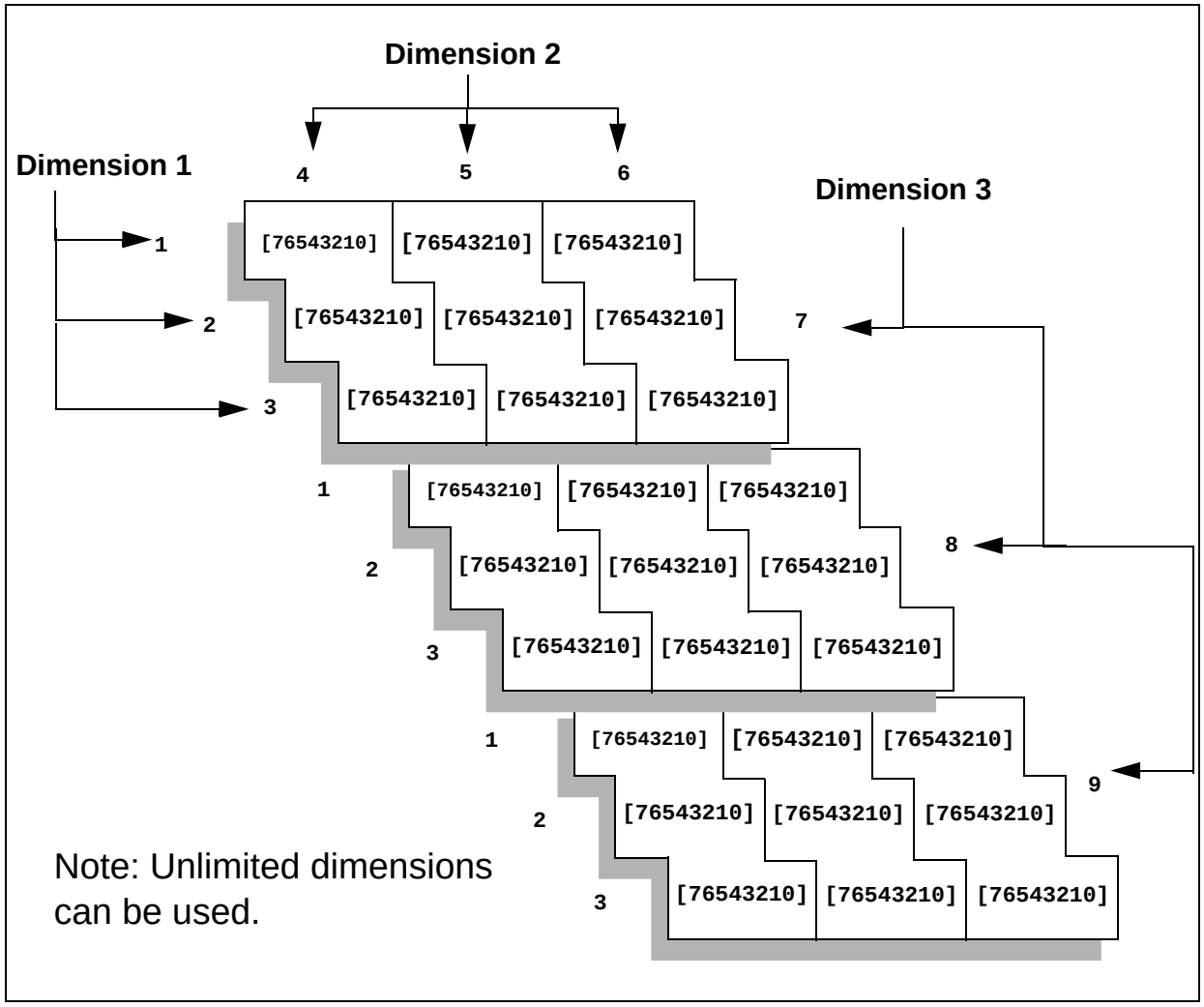
Note that `mem01` in the following example is a three-dimensional array. It has 3x3x3 (27) locations; each location is 8 bits in length.

```
reg [3:0] addr1L, addr1R, addr2L, addr2R, addr3L, addr3R;

reg [7:0] mem01 [1:3] [4:6] [7:9]
```

(See Figure 17-1 for an graphical representation of the previous example).

Figure 17-1 Diagram of example: `reg [7:0] mem01 [1:3] [4:6] [7:9]`

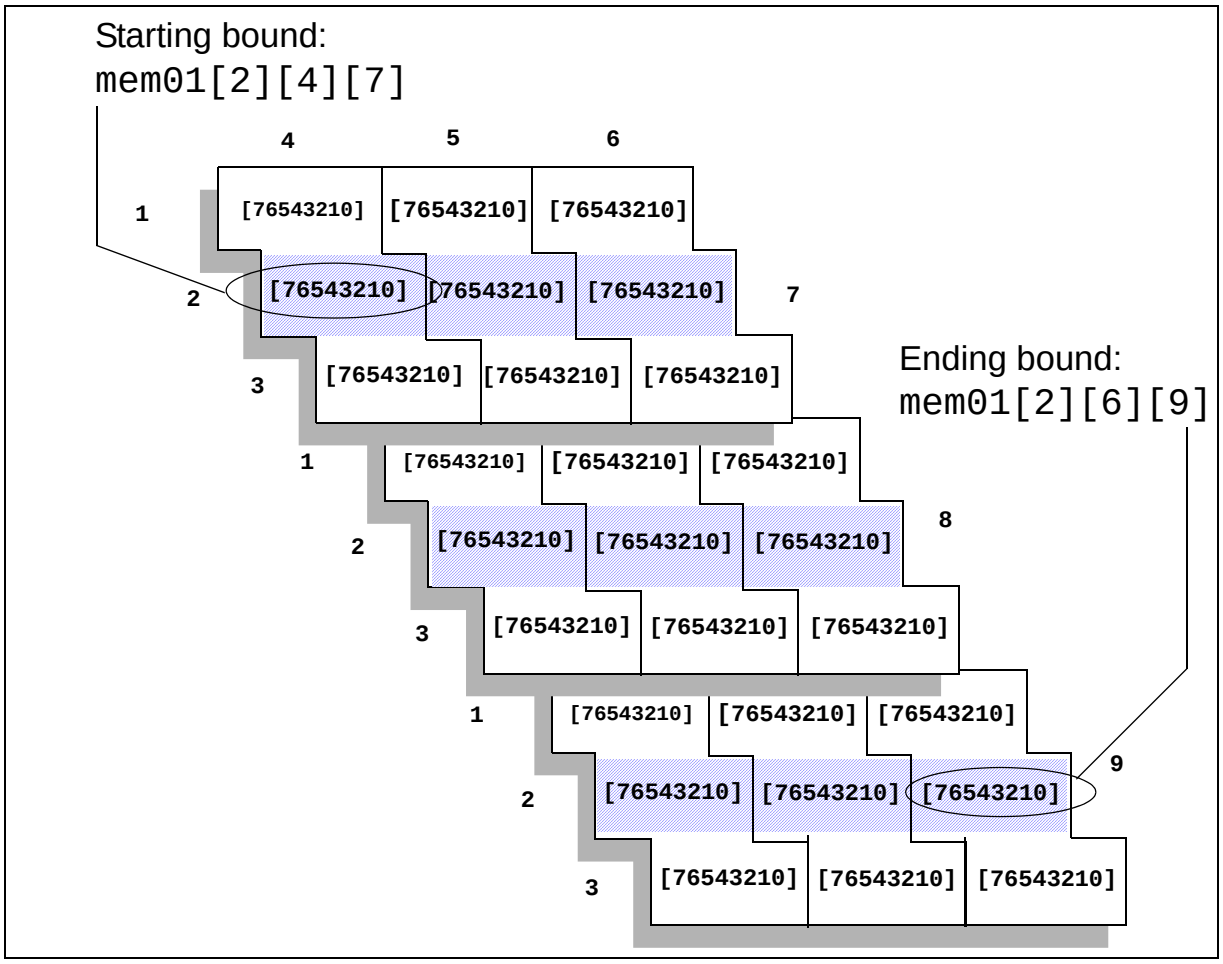


```
$vcdplusmemon( mem01 );
    // Records all elements of mem01 to the VPD file.


addr1L = 2;
$vcdplusmemon( mem01, addr1L );
// Records elements mem01[2][4][7] through mem01[2][6][9]
```

The elements highlighted by the in the diagram in Figure 17-2 demonstrate the previous example.

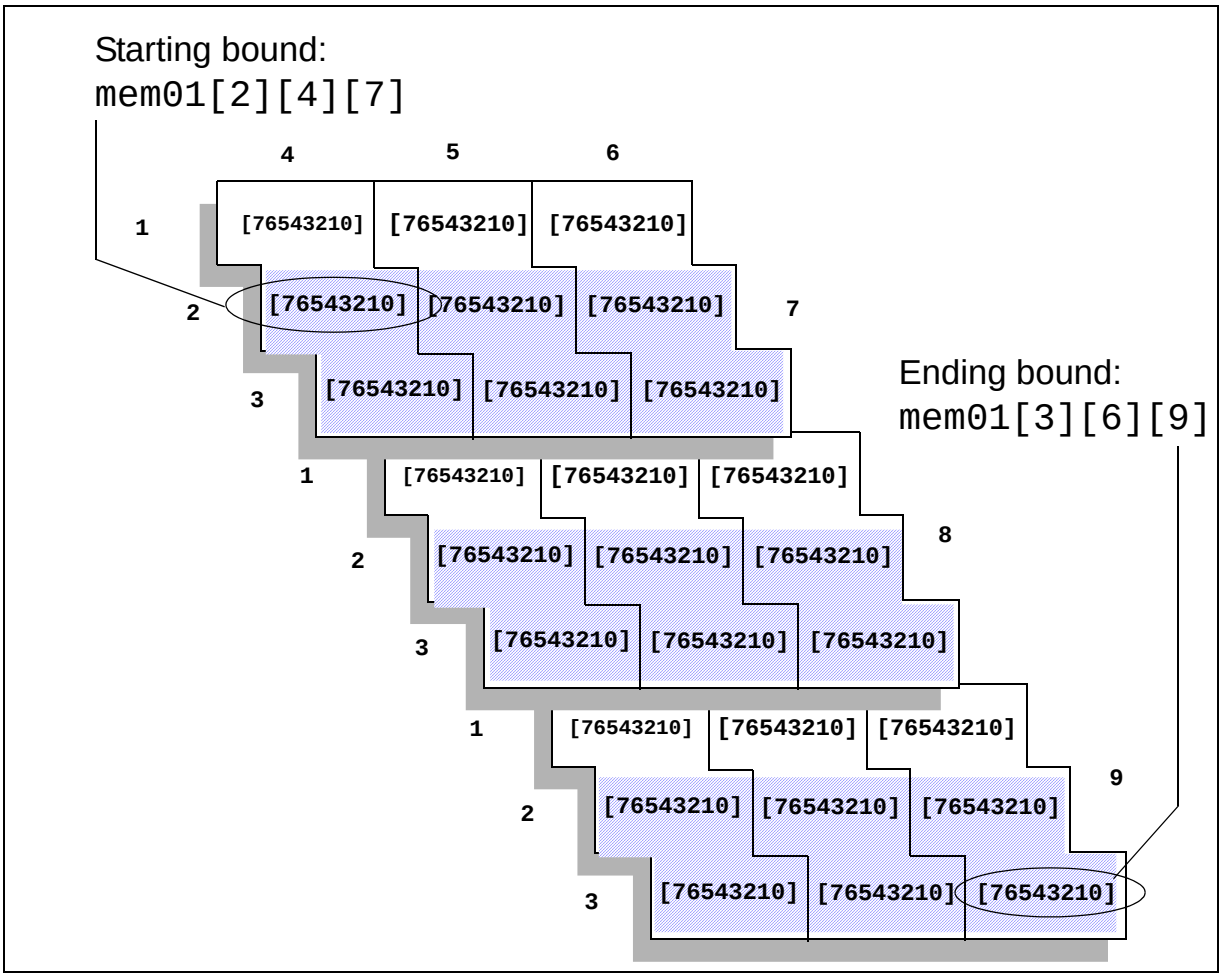
Figure 17-2 Diagram of example: `$vcdplusmemon( mem01, addr1L );`



```
addr1L = 2;
addr1R = 3;
$vcdplusmemon( mem01, addr1L, addr1R );
// Records elements mem01[2][4][7] through mem01[3][6][9]
```

The elements highlighted by the in the diagram in Figure 17-3 demonstrate the previous example.

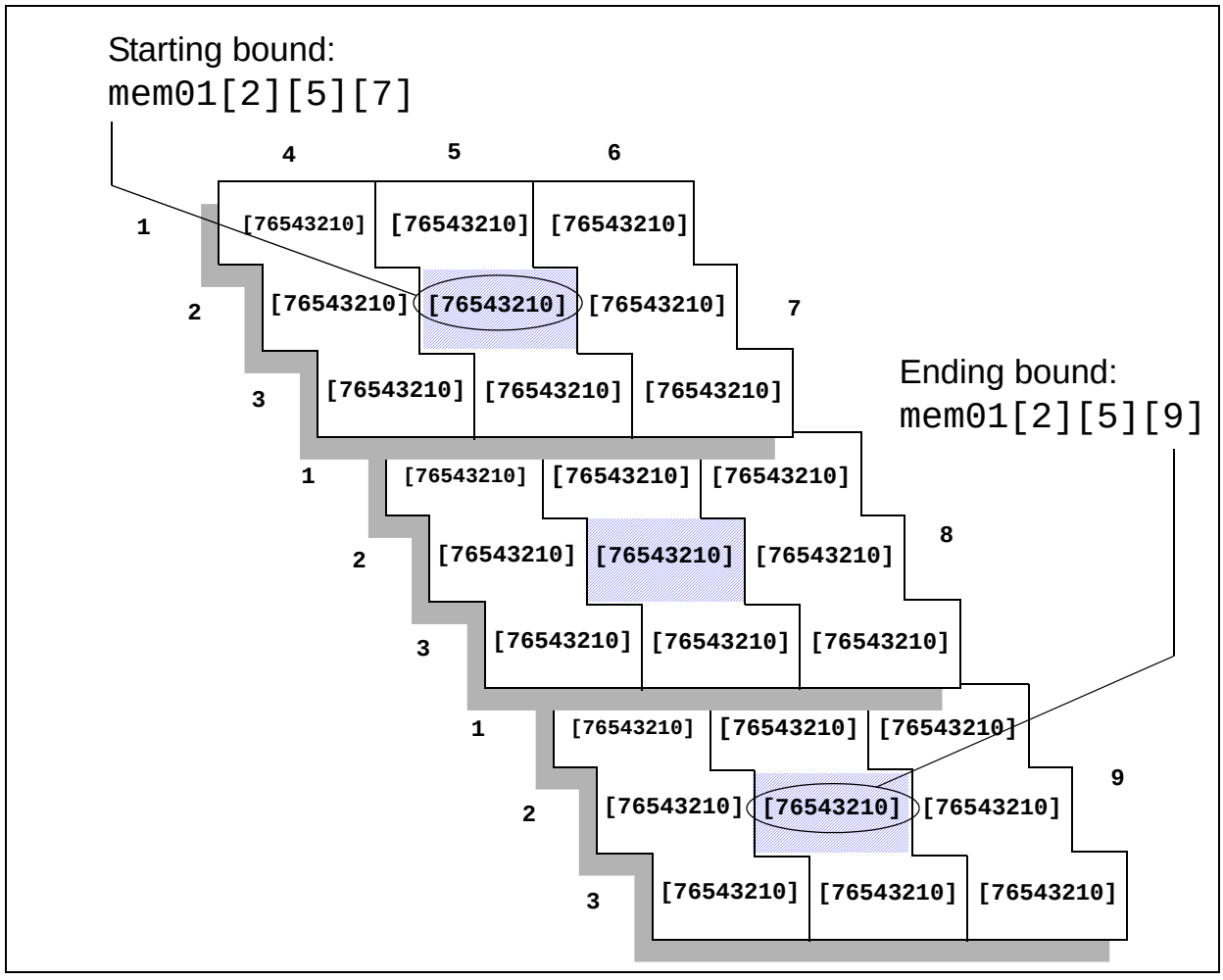
Figure 17-3 `$vcdplusmemon( mem01, addr1L, addr1R );`



```
addr1L = 2;
addr1R = 2;
addr2L = 5;
$vcdplusmemon( mem01, addr1L, addr1R, addr2L );
// Records elements mem01[2][5][7] through mem01[2][5][9]
```

The elements highlighted by the in the diagram in Figure 17-4 demonstrate the previous example.

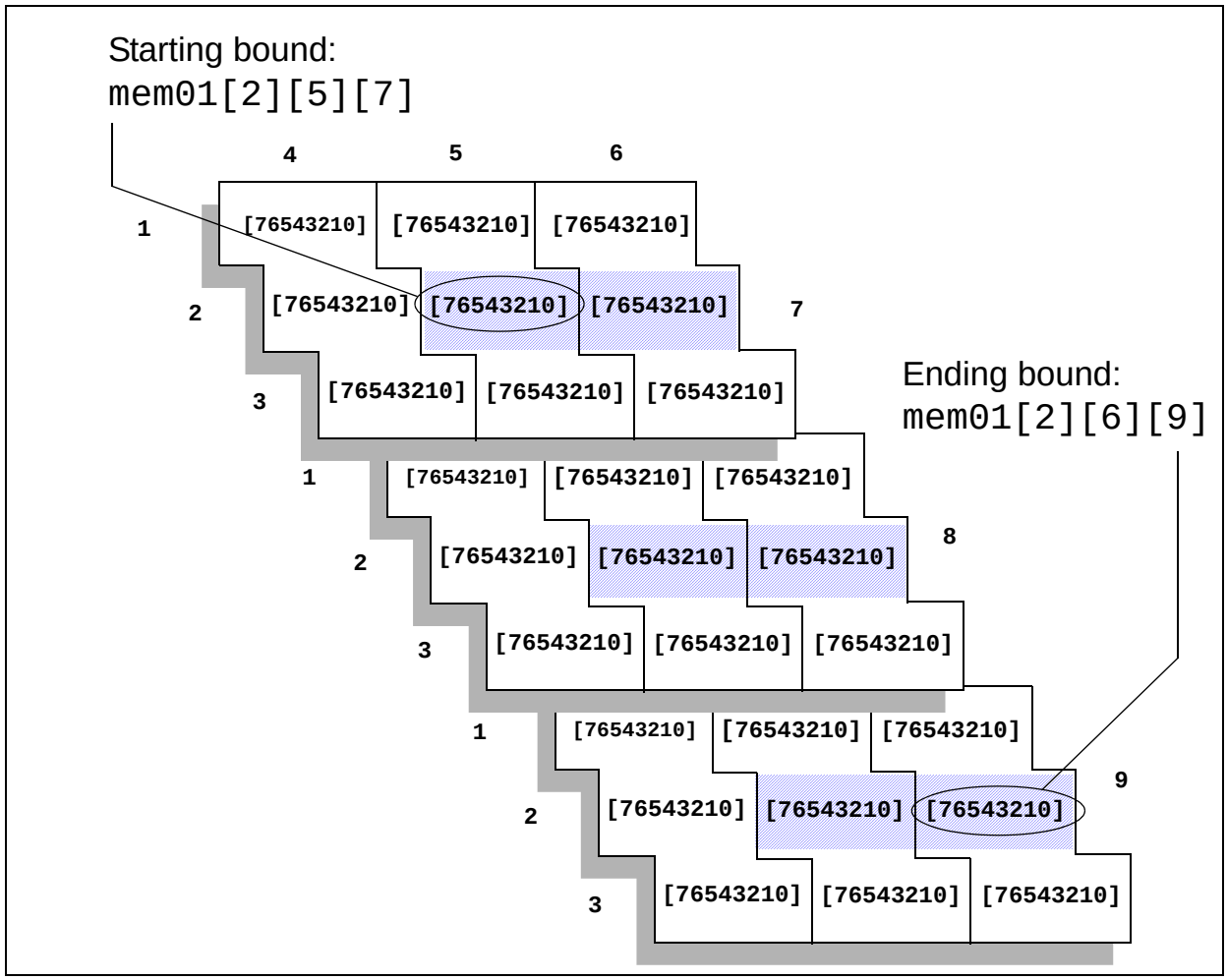
Figure 17-4 `$vcdplusmemon( mem01, addr1L, addr1R, addr2L );`



```
addr1L = 2;
addr1R = 2;
addr2L = 5;
addr2R = 6;
$vcdplusmemon( mem01, addr1L, addr1R, addr2L, addr2R );
// Records elements mem01[2][5][7] through mem01[2][6][9]
```

The elements highlighted by the in the diagram in Figure 17-5 demonstrate the previous example.

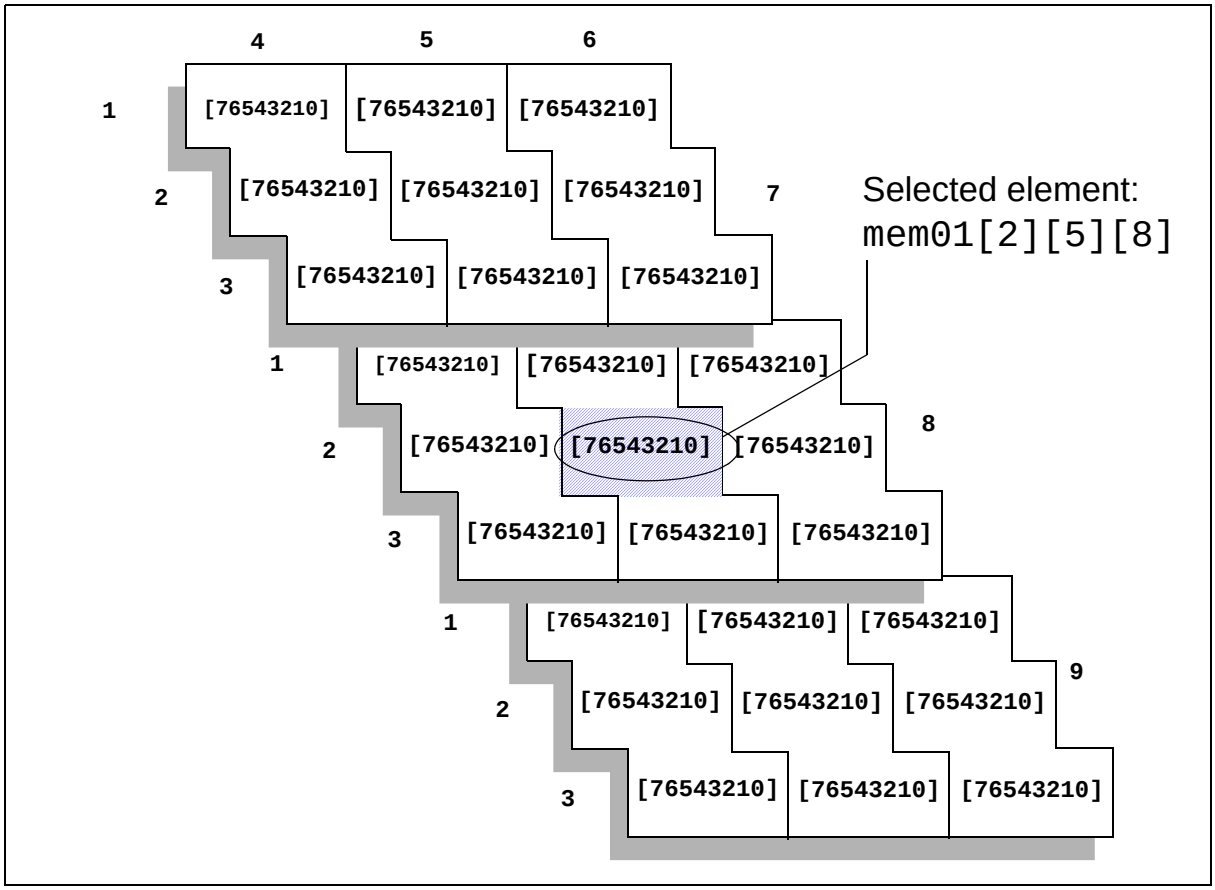
Figure 17-5 $vcdplusmemon(mem01, addr1L, addr1R, addr2L, addr2R);



```
addr1L = 2;
addr1R = 2;
addr2L = 5;
addr2R = 5;
addr3L = 8;
addr3R = 8;
$vcdplusmemon( mem01, addr1L, addr1R, addr2L, addr2R, addr3L
);
$vcdplusmemon( mem01, addr1L, addr1R, addr2L, addr2R,
addr3L, addr3R );
// Either command records element mem01[2][5][8]
```

The element highlighted by the in the diagram in Figure 17-6 demonstrates the previous example.

Figure 17-6 Selected element: `mem01[2][5][8]`



Using the $vcdplusmemorydump Task

The `$vcdplusmemorydump` task dumps a snapshot of memory locations. When the function is called, the current contents of the specified range of memory locations are recorded (dumped).

The complete set of multi-dimensional array elements to be dumped can be specified only once. You can specify multiple element subsets of an array using multiple `$vcdplusmemorydump` commands, but they must occur in the same simulation time. In subsequent simulation times, `$vcdplusmemorydump` commands must use the initial set of array elements or a subset of those elements. Dumping elements outside the initial specifications will cause a warning.

Within VirSim, multi-dimensional arrays can be expanded in each dimension in much the same way as memories. By default, only the portions of the multi-dimensional array that have data are shown in VirSim.

System Tasks for Capturing Source Statement Execution Data

(Verilog only) The Source Window requires the use of VCD+ options to save source hierarchy information and VCD+ tasks to capture source statement execution information in VCD+ files. Capturing source statement execution allows you to view and trace statement execution in the Source Window.

Note that saving statement execution data can significantly increase simulation time and VCD+ file size.

The following information is covered in this section:

- How to Capture Verilog Source Statement Execution

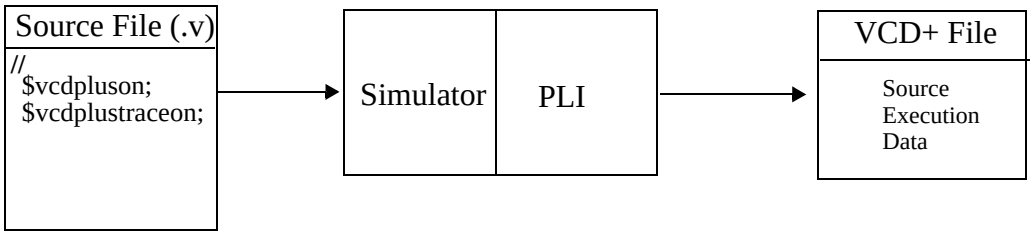
- Source Statement System Tasks

How to Capture Verilog Source Statement Execution

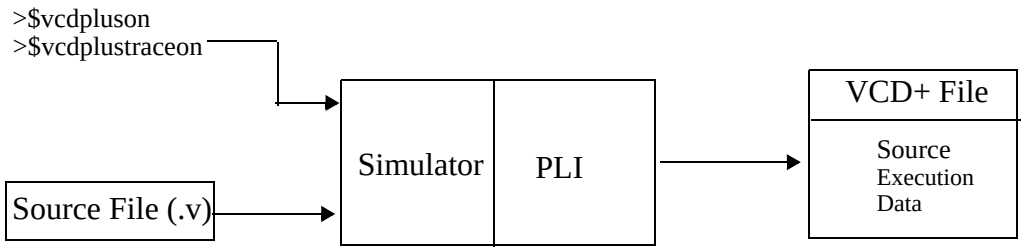
Figure 17-7 shows three ways to capture source statement execution.

Figure 17-7 Three Ways of Capturing Verilog Source Statement Execution

1. For viewing in post simulation mode, include **`$vcdplustraceon`.**



2. For viewing in post simulation mode, enter the appropriate trace task at the simulator command line.



3. For viewing in interactive mode in the VirSim Source Window, Capture Line Data must be enabled (enabled by default). To also generate a VCD+ file for viewing in post simulation mode, include the appropriate trace task in the source file.

Source Statement System Tasks

- $vcdplustraceon
- $vcdplustraceoff

Note:
For VCS you also must supply the `-line` option when creating the simulator executable.

$vcdplustraceon

The `$vcdplustraceon` task turns on line tracing. Line trace information is saved in the VCD+ file.

Syntax:
`$vcdplustraceon (<level>,<scope>*);`

where

level specifies the number of hierarchy scope levels to descend to record line tracing (a zero value records all line tracing to the end of the hierarchy; default is 1 level).
scope specifies the name of the scope in which to record line tracing (default is 1 level).

* indicates that argument can have a list of more than one value (for scopes).

$vcdplustraceoff

The `$vcdplustraceoff` task turns off line tracing.

Syntax:
`$vcdplustraceoff (<level>,<scope>*);`

where

level specifies the number of hierarchy scope levels to desend to stop recording line tracing (a zero value stops the recording of all line tracing to the end of the hierarchy; default is 1 level).

System Tasks for Capturing Delta Cycle Information

The following VCD+ system tasks are used to capture and display Delta Cycle information in the Waveform Window.

- $vcdplusdeltacycleon
- $vcdplusdeltacycleoff

$vcdplusdeltacycleon

This command enables reporting of Delta Cycle information from the simulator CLI or the Verilog source code. This must be followed by the appropriate `$vcdpluson`/`$vcdplusoff` commands.

Glitch detection is automatically turned on when $vcdplusdeltacycleon is executed unless you have previously used `$vcdplusglitchon/off`. Once you have used `$vcdplusglitchon/off`, VirSim allows you explicit control of glitch detection.

Syntax:

```
$vcdplusdeltacycleon;
```

Note:

Delta Cycle collection can start only at the beginning of a time sample. vcdplusdeltacycleon must precede the vcdpluson command to ensure that Delta Cycle collection will start at the beginning of the time sample.

$vcdplusdeltacycleoff

This command turns off reporting of Delta Cycle information starting at the next sample time.

Glitch detection is automatically turned off when `$vcdplusdeltacycleoff` is executed unless you have previously used `$vcdplusglitchon/off`. Once you have used `$vcdplusglitchon/off`, VirSim allows you explicit control of glitch detection.

Syntax:

```
$vcdplusdeltacycleoff;
```

System Tasks for Capturing Unique Event Information

The following VCD+ system tasks are used to capture unique events and glitch information.

- $vcdplusglitchon
- $vcdplusglitchoff
- $vcdplusevent

$vcdplusglitchon

The `$vcdplusglitchon` task turns on checking for zero delay glitches and other cases of multiple transitions for a signal in one sample time. Glitch detection is automatically turned on when `$vcdplusdeltacycleon` is executed unless you have previously used `$vcdplusglitchon/off`. Once you have used `$vcdplusglitchon/off`, VirSim allows you explicit control of glitch detection.

When a glitch is detected for a signal, a zero delay glitch event is recorded. The default setting is not to perform zero delay glitch detection.

Syntax:

```
$vcdplusglitchon;
```

$vcdplusglitchoff

The `$vcdplusglitchoff` task turns off checking for zero delay glitches. Glitch detection is automatically turned off when `$vcdplusdeltacycleoff` is executed unless you have previously used `$vcdplusglitchon/off`. Once you have used `$vcdplusglitchon/off`, VirSim allows you explicit control of glitch detection.

Syntax:

```
$vcdplusglitchoff;
```

$vcdplusevent

The `$vcdplusevent` task allows the user to record a unique event for a signal at the current simulation time unit. These events can be displayed in the Waveform Window, Logic Browser, and Register Window.

There can be a maximum of 244 unique events, plus the pre-defined "glitch" event which is automatically generated by the pli, and a "Too many events" event which all unique events beyond the allowed 244 will be automatically named.

Syntax:

```
$vcdplusevent (<signal>, "<event_name>",
           "<severity><shape>");
```

where

signal is any valid signal name.
event_name is a unique string which describes the event. This event_name will appear in the status bar of the Waveform Window, Logic Browser, or Register Window when the mouse is placed on the event marker.
severity is a single character with legal values `E`, `W`, or `I`, which indicates the severity of the event. The severity of the event may be Error, Warning, or Information respectively. Colors associated with the severity level are set in the X Resource file or NT registry. The defaults are Red=Error, Yellow=Warning, and Green=Information. If the severity is not interpretable, it will default to `E`.
shape is a single character with legal values `S`, `T`, or `D` which indicates the geometry of the event as drawn by Virsim, and are Square, Triangle, and Diamond respectively. If the geometry is not interpretable, it will default to `T`.

Simulator Run-Time Options

Specific command line options are used to generate VCD+. These options are used to set the RAM buffer size, provide the VCD+ default file name, specify the VCD+ file size, ignore file calls, check licenses, and control what information is stored. This section describes the following commands:

- +vpdbufsize to control RAM Buffer Size
- +vpdfile to set the output file name
- +vpdfilesize to control maximum file size
- +vpdignore to ignore $vcdplus calls in code
- +vpddrivers to store driver information
- +vpdports to store port information
- +vpdnocompress to bypass data compression

+vpdbufsize to control RAM Buffer Size

To gain efficiency, VCD+ uses an internal buffer to store value changes before saving them on disk. The `+vpdbufsize` command modifies the size of that internal buffer. The minimum size allowed is what is required to share two value changes per signal.

Syntax:
`+vpdbufsize+`*nn*

Where *nn* is buffer size in megabytes (default is the size required to store 15 value changes for each signal but not less than 2 megabytes).

Note:

The buffer size automatically is increased as needed to comply with the above limit.

+vpdfile to set the output file name

The vpdfile command allows specification of the output file name.

Syntax:

`+vpdfile+`*xxx*

Where *xxx* is the VCD+ filename (default is `vcdplus.vpd`). You must include the full file name with the `.vpd` extension.

+vpdfilesize to control maximum file size

The `+vpdfilesize` command creates a VCD+ file, which has a moving window in time while never exceeding a specified file size *nn megabytes*. When the VCD+ file size limit is reached, VCD+ will continue saving simulation history by overwriting older history.

File size is a direct result of circuit size, circuit activity, and the data being saved. Test cases show that VCD+ file sizes will likely run from a few megabytes to a few hundred megabytes. Many VirSim users can share the same VCD+ history file, which may be a reason for saving all time value changes when you do simulation. You can save one history file for a design and overwrite it on each subsequent run.

Syntax:

`+vpdfilesize+`*nn*

Where *nn* is the file size in megabytes.

+vpdignore to ignore $vcdplus calls in code

The `+vpdignore` command instructs the simulator to ignore any `$vcdplusxx` calls and license checking. By default, the simulator checks out a VCD+ PLI license if there is a `$vcdplusxx` in the Verilog source. In some cases, this statement is never executed and VCD+ PLI license checkout should be suppressed. The `+vpdignore` command performs the license suppression.

Syntax:
`+vpdignore`

+vpddrivers to store driver information

By default, VCD+ records value changes only for the resolved value for each net. To also report value changes for all its drivers when there are more than one driver, use the +vdpdrivers option when simulating. The driver values, for example, enable the Logic Browser to identify which drivers produce an undesired X on the resolved net.

This option affects performance and memory usage for larger designs or longer runs.

Syntax:
`+vpddrivers`

+vpdports to store port information

By default, VCD+ does not store the port type for each signal, and the Hierarchy Browser views all signals as internal and not connected to a port.

The +vpdports option causes VCD+ to store port information, which is then used by the Hierarchy Browser to show whether a signal is a port and if so its direction. This option to some extent affects simulation initialization time and memory usage for larger designs.

Syntax:
`+vpdports`

+vpdnocompress to bypass data compression

By default, VCD+ compresses data as it is written to the VCD+ file. The user may disable this feature by supplying the `+vpdnocompress` command line option.

Syntax:
`+vpdnocompress+`

vpdnostrengths to not store strength information

By default, VCD+ stores strength information on value changes to the VCD+ file. Use of this option may lead to slight improvements in simulator performance.

Syntax:
`+vpdnostrengths`

VCD+ Methodology

The following information explains how to manage the VirSim and VCD+ functions and features to optimize simulation and analysis.

- Advantages of Separating Simulation from Analysis
- Conceptual Example of Using Verilog VCD+ System Tasks
- VCD+ On/Off PLI Rules
- Performance Tips

Advantages of Separating Simulation from Analysis

When a problem is debugged, traditionally, interactive debugging has required a user to occupy one simulator license while simulating, thinking, resimulating, thinking...

Simulating once and efficiently storing as much data as possible allows for a more efficient debug methodology:

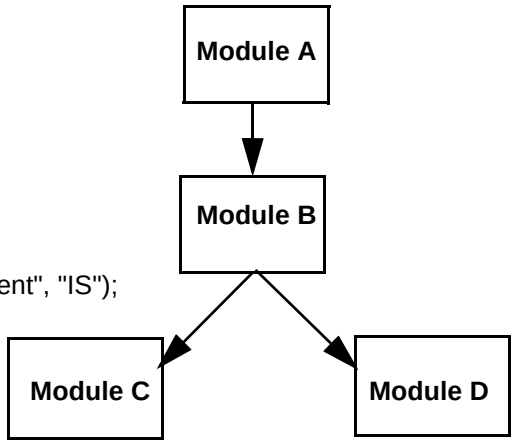
- The simulator is used once and then released to others.
- The analysis tool can go both forwards and backwards in time and analyze the complete set of data.
- The same set of data can be used by multiple engineers to debug one or more problems in parallel.

Conceptual Example of Using Verilog VCD+ System Tasks

The example in Figure 17-8, Example Definition of VCD+ Signal Capture (Recording), shows the entry of the `$vcdplus` system tasks in Module B scope. The dump saves all the variables in Module B from time 100 to 300, all variables in module C from time 200 to 500, and a single variable in module D1.clk from time 600 to 900. Zero delay glitch detection is on while value change data is being recorded throughout the simulation. Delta cycle information is stored starting at the first value change that occurs after time 200 and ending after the last value change during time 300. At time 700 a unique event is added to signal `D.clk`.

Figure 17-8 Example Definition of VCD+ Signal Capture (Recording)

```
fork
        #0 $vcdplusglitchon;
        #100 $vcdpluson (1,B);
        #200 $vcdplusdeltacycleon;
        #200 $vcdpluson (1,C);
        #300 $vcdplusdeltacycleoff;
        #300 $vcdplusoff (1,B);
        #500 $vcdplusoff (1,C);
        #600 $vcdpluson (D.clk);
        #700 $vcdplusevent (D.clk, "UserEvent", "IS");
        #900 $vcdplusoff (D.clk);
join
```



Methods

You can implement signal data capture (recording) control in the source code and at the shell command line, as shown in the following Verilog examples.

- Create a task in source:

```
task sigon_instreg;
 begin
  $vcdpluson(test.risc1.instreg);
 end
endtask
```

  Then call the task from source code.

```
initial
 sigon_instreg;
```

  Or, enter the task name at the simulator interactive prompt.

```
C1> sigon_instreg;
```

- Using a shell command argument to enable task execution:

```
vcs -f run.f +signal_on

 initial
  if ($test$plusargs("signal_on"))
   sigon_instreg;
 task sigon_instrg;
  begin
   $vcdpluson(test.risc1.instreg);
  end
 endtask
```

VCD+ On/Off PLI Rules

- VCD+ On/Off PLI system tasks follow these basic rules.
- `vcdpluson` and `vcdplusoff` tasks may be inserted in source code or entered at the simulator interactive prompt.
- `vcdpluson` and `vcdplusoff` tasks accept one level but multiple scopes/signals as arguments.
- `vcdpluson` and `vcdplusoff` tasks when applied to the same signal(s) toggle the recording on and off. The count for each signal is accumulative; (+,-). ..on/..on/..off leaves the signal recording on. For example using the hierarchy of Figure 17-8, the following command sequence will still report on Module D since it was added twice but only removed once.

  ```
  $vcdpluson(A); $vcdpluson(B); $vcdplusoff(D);
  ```

- On large designs, you should selectively turn signal data capture (dumping) on or off. Multiple use of `vcdpluson` and `vcdplusoff` allow on and off selection
- `vcdpluson` and `vcdplusoff` tasks executed in the same simulation time period may execute in any order. To ensure that one or the other executes last, separate them by at least one simulation time unit.
- Signals that are turned off may have signal value changes recorded if a higher/ lower level of the same signal is turned on.

Performance Tips

The following tips explain how to manage performance of the simulator and VCD+.

- Normally you should save data for all signals that you may require for analysis. The time range should be such that it very likely contains the origination of the problem.
- Generally, the bigger you make the RAM buffer size (via the `+vpdbufsize` option), the faster the simulation will complete. The effects are so dependent on circuit and activity that rules-of-thumb do not apply. We suggest doubling the RAM size for the same simulation on your own design while measuring simulator performance to get a figure for an appropriate setting. When making this measurement, compare sizeable simulation runs to overcome the effects of compile time. Naturally, the above requires that you have physical memory to accommodate the simulation. Swapping will significantly reduce performance.

  Making the buffer size too large can cause excessive swapping, which can dramatically slow the simulation.
- Saving line-execution data enables more efficient debug of misbehavior in behavioral code. Access to such data allows breakpointing on particular activities in the code and stepping through the exact execution of the source. Correctly used, the cost of reporting on line execution data (slower simulation) will more than pay for itself by enabling much faster location of the code deficiencies

- Saving statement execution for an entire design can increase simulation time by eight times or more. To limit performance degradation, limit the use of statement saves to certain scopes. Instead of saving statement execution from time 0, turn on tracing just prior to the time of a suspected problem and off after that time.
- The file size increases from 200 to 500 percent when saving line execution data.
- Glitch detection and delta cycle may significantly increase the size of the vpd file.
- The option `+vpdports` costs some CPU time and memory in the initialization. It allows telling ports from internal signals in the hierarchy browser.
- The option `+vpddrivers` costs some CPU time and memory during the simulation. However, it allows visibility to the individual values of drivers of a multiply driven net.
- The option `+vpdfilesize` (file wrap) can be used in a verification environment where stimuli is automatically generated or read and the results are verified by the test bench. Then, the simulation can be made to stop when an error occurs, and one is guaranteed to have the required history data even in a relatively small file that is left.

18

Translating VCD and VCD+

This section covers two optional utilities: vcd2vpd and vpd2vcd. These utilities convert VCD or EVCD files to VCD+ files or VCD+ files to VCD files. VirSim ships with a built in vcd2vpd utility so one could open a VCD file from within VirSim.

Note:
These two utilities apply to Verilog only.

vcd2vpd Command

The command performs one of the following operations:

- Converts a standard VCD file to a VCD+ file (`.vpd` extension).
- Converts an LSI Logic EVCD file with *$port* definitions and associated data to a VCD+ file. For information on the mapping of EVCD signals to VCD+ signals, see Table 18-1, Table 18-2, and Table 18-3.

Syntax

`vcd2vpd <options> <evcd_options> <vcd_file or evcd_file> <vpd_file>`

Options

-b#	The buffer size in kilobytes(#) to use to store value change data before writing the data to disk.
-f#	The file size in kilobytes (#). Wrap around occurs if the specified file size is exceeded.
-h	Translate hierarchy information only.
-m	Give translation metrics during the conversion.
+deltacycle	Add delta cycle information to each signal value change.

EVCD Options

+dut+<*dt_prefix*>	Modifies the string identifier for the Device-Under-Test (DUT) half of the split signal. The default name is DUT.
+tf+<*tf_prefix*>	Modifies the string identifier for the Test Fixture (TF) half of the split signal. The default name is TF.
+indexlast	Appends the bit index of a vector bit as the last element of a name. In SVCD translation, an EVCD signal is split into two new signals: one signal for the test-fixture (TF) value and one signal for the Device-Under-Test (DUT) value. For example: • foo_DUT_1 • foo_DUT_2 • foo_TF_1 • foo_TF_2 In the default (if the `+indexlast` option is omitted), the signals are split in two and organized by the index number: • foo_1_DUT • foo_1_TF • foo_2_DUT • foo_2_TF

Files

vcd_file or evcd_file	The VCD source file that is converted to the VCD+ file.
vpd_file	The VCD+

Mapping EVCD to VCD+ Signals

In the conversion, each EVCD port maps to two new VCD+ signals: a Test Fixture (TF) signal and a Device-Under-Test (DUT) signal. Table 18-1, Table 18-2, and Table 18-3 show the mapping of values in the EVCD file to values in the VCD+ file.

Table 18-1 Mapping In Input Mode (TF Drives)

EVCD Value Maps	To VCD+ TF	To VCD+ DUT
D low	St0	HiZ
U high	St1	HiZ
N unknown	StX	HiZ
Z tri-state	SmZ	HiZ
d low (2 or more drivers active)	Su0	HiZ
u high (2 or more drivers active)	Su1	HiZ

Table 18-2 Mapping In Output Mode (DUT Drives)

EVCD Value Maps	To VCD+ TF	To VCD+ DUT
L low	HiZ	St0
H high	HiZ	St1
X unknown (value not important)	HiZ	Stx
T tri-state	HiZ	SmZ
l low (2 or more drivers active)	HiZ	Su0
h high (2 or more drivers active)	HiZ	Su1

Table 18-3 Mapping in Bidirectional Mode

EVCD Value Maps	To VCD+ TF	To VCD+ DUT
0 low (Both TF and DUT active with 0 value)	St0	St0
1 high (both TF and DUT active with 1 value)	St1	St1
? unknown	StX	StX

Table 18-3 Mapping in Bidirectional Mode (Continued)

EVCD Value Maps	To VCD+ TF	To VCD+ DUT
F tr-state (TF and DUT unconnected)	HiZ	HiZ
A unknown (TF 0 and DUT 1)	St0	St0
a unknown (TF 0 and DUT x)	St0	StX
B unknown (TF 0 and DUT 0)	St1	St0
b unknown (TF 1 and DUTX)	St1	Stx
C unknown (TF X and DUT 0)	StX	St0
c unknown (TF X and DUT 1)	Stx	St1
f unknown (TF and DUT tri-stated)	Smz	Smz

vpd2vcd Command

The vpd2vcd command converts the binary VCD+ file to the standard VCD file in ASCII format.

Syntax

```
vpd2vcd <options> <vpd_file> <vcd_file>
```

Options

-h	Translate hierarchy information only
-m	Give translation metrics during the conversion
-s	Allow sign extension for vectors. Reduces size of `<vcd.file>`
+morevhdl	Translate VHDL types that are not directly mappable to verilog types. These VHDL types are in addition to the ones that are directly mappable. See VHDL Mapping.
+ignoredelta	Leave delta cycle information out of the translation. This improves efficiency when the VPD file contains delta cycle information that you do not need in the VCD file.
+start+<value>	Translate value changes starting after start time <*value*>.
+end+<value>	Translate value changes ending before end time <*value*>.

Files

vpd_file	The VCD+ file that is to be converted VCD file.
vcd_file	The VCD file generated from the VCD+ file.

VHDL Mapping

The following VHDL constructs are mapped for translation from VPD to VCD. Additional VHDL constructs can be pseudo mapped using the +morevhdl switch (see VHDL Mapping). Using the +morevhdl switch allows you to view the additional VHDL constructs, but there may be inaccuracies in the pseudo mapped signals, and you cannot translate a VCD file that uses pseudo mapping back into a VPD file.

Table 18-4 Mapping of VHDL Constructs for Translation to VCD

VHDL Construct	**vpd2vcd Mapping**
package	module
entity	module, task
process	function (always or initial statement)
block	begin
bit	wire
Std_logic n	wire n/4 (n is an integer divisible by 4)
Integer	integer
real	real
time	time
boolean	wire
character	wire 8
enum	integer
arrays	wire/integer/real

19

Viewing OpenVera Assertions

This chapter introduces OpenVera Assertions and explains how to view them in VirSim.

The following topics are covered in this chapter:

- Introducing OpenVera Assertions
- How Sequences Are Tested
- Viewing OVA Results in VirSim

Introducing OpenVera Assertions

OpenVera Assertions (OVA) provides a clear, easy way to describe sequences of events and facilities to test for their occurrence. With clear definitions and less code, testbench design is faster and easier. And you can be confident that you are testing the right sequences in the right way.

OpenVera Assertions is a declarative method that is much more concise and easier to read than the procedural descriptions provided by hardware description languages such as Verilog. With OpenVera Assertions:

- Descriptions can range from the most simple to the most complex logical and conditional combinations.
- Sequences can specify precise timing or a range of times.
- Descriptions can be associated with specified modules and module instances.
- Descriptions can be grouped as a library for repeated use. OpenVera Assertions includes a Checker Library of commonly used descriptions.

Built-in Test Facilities and Functions

OpenVera Assertions has built-in test facilities to minimize the amount of code that you need to write. In addition, OpenVera Assertions works seamlessly with other Synopsys tools to form a complete verification environment. OpenVera Assertions:

- Tests Verilog, VHDL, and mixed-HDL designs using the VCS and Scirocco simulators.
- Automatically tests and reports results on all defined sequences. You just write the definitions.
- Produces results that can be viewed with VirSim.
- Can be monitored and controlled as part of a Vera testbench.

How Sequences Are Tested

Testing starts with a *temporal assertion file*, which contains the descriptions of the sequences and instructions for how they should be tested. OpenVera Assertions is designed to resemble Verilog with similar data types, operators, and lexical conventions.

Example 19-1 shows an example temporal assertion file. It tests for a simple sequence of values (4, 6, 9, 3) on the device's outp bus.

Example 19-1 Temporal Assertion File, cnt.ova

```
/* Define a unit with expressions and assertions (or select
one from the Checker Library).
*/
unit 4step
  #(parameter integer s0 = 0)      // Define parameters
  (logic clk, logic [7:0] result); // Define ports

  // Define a clock to synchronize attempts:
  clock posedge (clk)
  {
    // Define expressions:
    event t_0 : (result == s0);
    event t_1 : (result == 6);
    event t_2 : (result == 9);
    event t_3 : (result == 3);
    event t_normal_s: t_0 #1 t_1 #1 t_2 #1 t_3;
  }
```

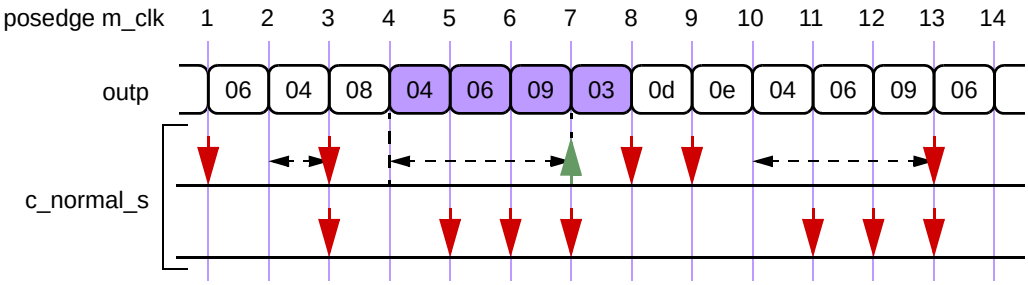
```
  // Define an assertion:
  assert c_normal_s : check(t_normal_s, "Missed a step.");

endunit

/* Bind the unit to one or more instances in the design.
*/
// bind module cnt : // All instances of cnt or
bind instances cnt_top.dut : // one instance.
  4step start_4   // Name the unit instance.
  #(4)            // Specify parameters.
  (m_clk, outp); // Specify ports.
```

When the temporal assertion file is compiled and run with a simulator, the assertions are continuously tested for the duration of the simulation. New attempts to match each assertion to the simulation's values are started with every cycle of the assertion's associated clock. Each attempt continues until it either fails to match or succeeds in matching the complete expression. See Figure 19-2. The up arrow at clock tick 7 indicates a match that started at tick 4. The down arrows are failures. The failure or success of each attempt is logged to a file that can be reviewed later.

Figure 19-2 Assertion Attempts for cnt.ova



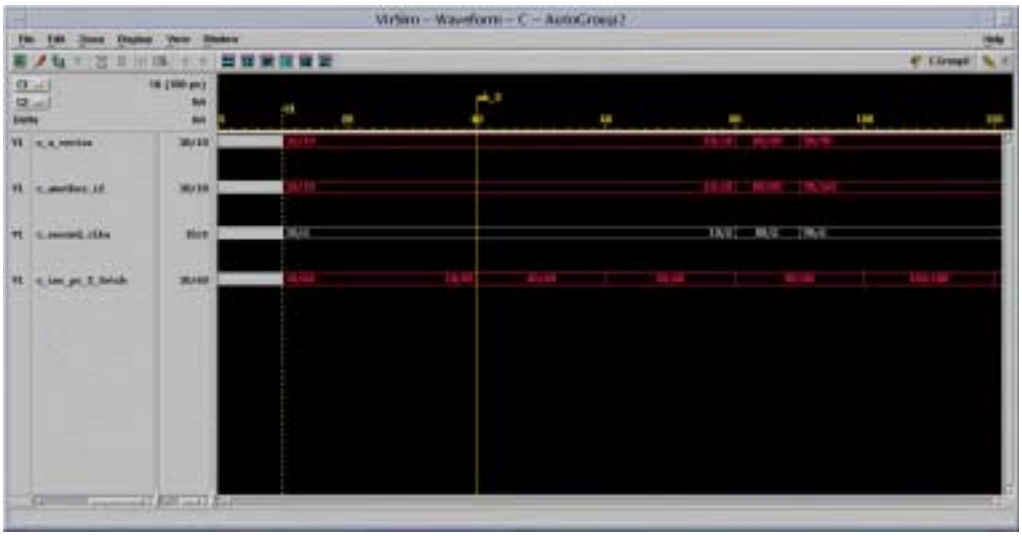
A Vera testbench can monitor and control the testing. Using built-in object classes, you can stop and start attempts to match the selected assertion; monitor attempts, failures, and successes; and synchronize the testbench with the testing process.

Viewing OVA Results in VirSim

You can view OVA results in VirSIm from a VCS, Scirocco, or Mixed-HDL simulation run if a design is compiled with the `-ova_debug` or `-ova_debug_vpd` option.

To view the assertion results, start VirSim and load the VPD file from the simulation run. Then go to the scope where the assertion is declared. The scope shows the assertion plus design signals and OVA variables related to the assertion. Drag the assertion and any other signals of interest into the Waveform Viewer. Use the features of VirSim to debug the results in the same way as debugging other simulation results. Figure 19-3 shows the Waveform Viewer with four assertions.

Figure 19-3 VirSim Display of Assertion Results



An assertion shows each evaluation attempt as a colored box: green for success, red for failure, gray for incomplete. A solid, gray bar from the beginning of the simulation time indicates that no attempts have started yet. The left edge of the box marks the start time of the attempt. However, because attempts can overlap, the right edge is not meaningful and the length of the box does not represent the length of the sequence. (The right edge is just the beginning of the next attempt.)

The start and end times are noted in the box as start/end. An end time of “X” means incomplete. An asterisk (*) means the left-most characters were truncated for lack of space. An empty box is also because of lack of space. To see the times, zoom in.

Expanding an assertion displays three component “signals” (see Figure 19-4):

- “clk” shows the ticks of the clock used by the assertion.
- “result” shows the result of each attempt: a green up arrow for success, a red down arrow for failure, a gray line for incomplete. The result markers are placed at the start times of the attempts.
- “end_time” notes the end time for each attempt. An “X” means incomplete.

Figure 19-4 VirSim Display with Expanded Assertion Results

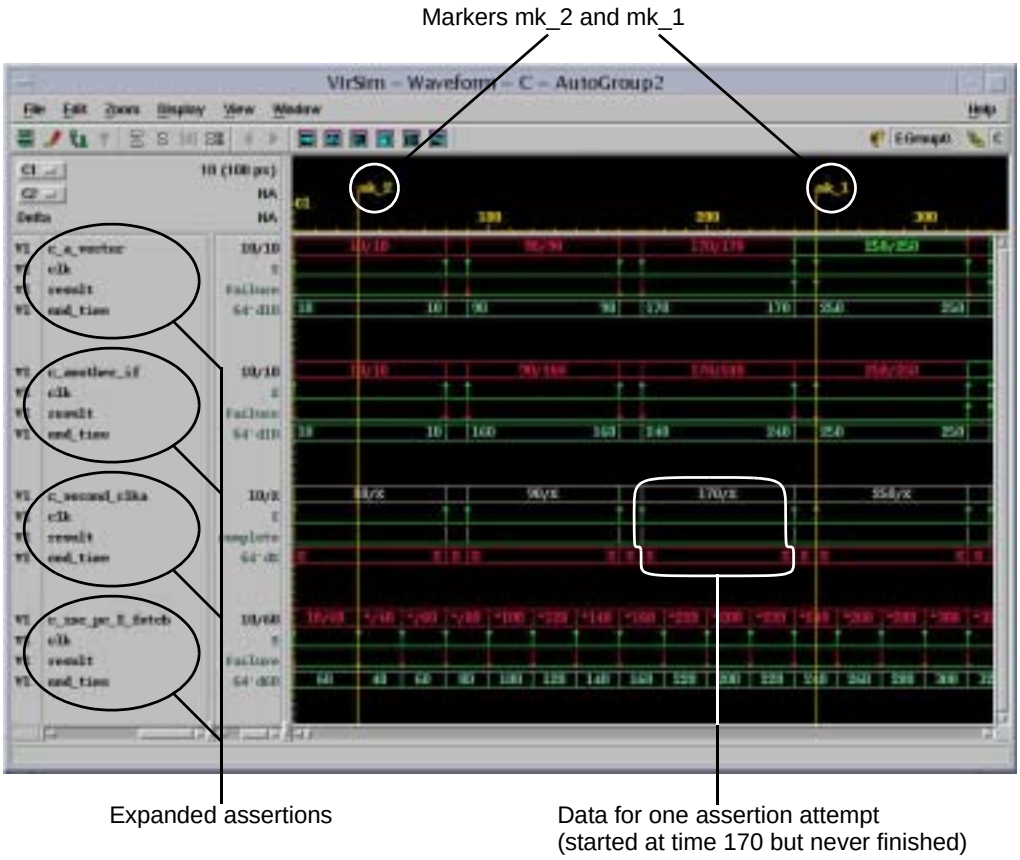


Figure 19-4 shows four assertions that have been expanded. The vertical markers, mk_1 and mk_2, highlight typical display elements.

Marker mk_1 (at time tick 250) is at the beginning of an attempt on each of the first three assertions. Assertion c_a_vector shows a success that started and ended at 250. Assertion c_another_if shows a failure that started and ended at 250. Assertion c_second_clka shows an incomplete attempt that started at 250 and never ended.

Assertion c_inc_pc_2_fetch, at marker mk_2, shows a display of overlapping attempts. The first box for the assertion shows an attempt that started at 10 and ended at 60. The second box shows an attempt that started and ended at 40—after the first attempt started but before it ended.

Assertion c_inc_pc_2_fetch also shows truncated start/end times marked with asterisks. You can also see that this assertion was evaluated with a different clock than the other three assertions.

A

Waveform Defaults

Waveform Style: Verilog Defaults

Table A-1, Verilog Waveform Defaults, describes the waveform display characteristics of Verilog signals in the Waveform Window. The Waveform Style Editor lets you assign waveform characteristics to a configuration.

When setting Verilog styles, consider how the following miscellaneous Waveform display characteristics apply to Verilog.

- Vector values are displayed inside the signal confines. If the zoom factor is too small to display the entire value, VirSim truncates as many characters as necessary and inserts an asterisk (*) on the truncated side of the value. If there is not enough room for at least two characters, no value is displayed. The values at C1 are displayed in the Signal Value pane.
- Named events are represented by single vertical arrows at the time the event occurs.
- A full gray bar indicates that no data is present for the signals at the time period displayed. During interactive simulation, the display is gray and updates as time advances. Since VCD+ allows you to turn signals on and off during simulation, signal displays may turn gray for a time period, then revert to normal logic values.
- Each waveform shows values in the radix enumeration selected for the waveform. Radices are selected from the Radix Menu

Table A-1 Verilog Waveform Defaults

Signal Value	Encoding Type	Default Color	Waveform Display	Resource Variable
Small 0	Scalar	Blue	line at bottom	eDsiColor13
Medium 0	Scalar	Blue	line at bottom	eDsiColor13
Weak 0	Scalar	Blue	line at bottom	eDsiColor13

Table A-1 Verilog Waveform Defaults (Continued)

Signal Value	Encoding Type	Default Color	Waveform Display	Resource Variable
Large 0	Scalar	Blue	line at bottom	eDsiColor13
Pull 0	Scalar	Blue	line at bottom	eDsiColor13
Strong 0	Scalar	Blue	line at bottom	eDsiColor13
Supply 9	Scalar	Blue	line at bottom	eDsiColor13
Small 1	Scalar	Green	line at top	eDsiColor18
Medium 1	Scalar	Green	line at top	eDsiColor18
Weak 1	Scalar	Green	line at top	eDsiColor18
Large 1	Scalar	Green	line at top	eDsiColor18
Pull 1	Scalar	Green	line at top	eDsiColor18
Strong 1	Scalar	Green	line at top	eDsiColor18
Supply 1	Scalar	Green	line at top	eDsiColor18
Small X	Scalar	Red	filled box	eDsiColor23
Medium X	Scalar	Red	filled box	eDsiColor23
Weak X	Scalar	Red	filled box	eDsiColor23
Large X	Scalar	Red	filled box	eDsiColor23
Pull X	Scalar	Red	filled box	eDsiColor23
Strong X	Scalar	Red	filled box	eDsiColor23
Supply X	Scalar	Red	filled box	eDsiColor23
High Z	Scalar	Gold	line at midpoint	dsiColor06
Range	Scalar	Yellow	open box with value	dsiColor68

Table A-1 Verilog Waveform Defaults (Continued)

Signal Value	Encoding Type	Default Color	Waveform Display	Resource Variable
Data	Vector	Aqua	open box with value	dsiColor14
Some X	Vector	Red	open box with value	dsiColor23
Some Z	Vector	Gold	open box with value	dsiColor06
All X	Vector	Red	filled box	dsiColor23
All Z	Vector	Gold	line at midpoint	dsiColor06
Any Value	Real	Cyan	open box with value	dsiColor15
Any Event	Event	Green	vertical arrow	dsiColor18
All X	Memory	Red	filled box	dsiColor23
Some X	Memory	Red	open box with value	dsiColor23
All Z	Vector	Gold	line at midpoint	dsiCoior06
Some Z	Vector	Aqua	open box with value	dsiColor14
Data	Vector	Aqua	open box with value	dsiColor14

Waveform Style: VHDL Defaults

Table A-2, VHDL Defaults, describes the waveform display characteristics of VHDL signals in the Waveform Window. With the Waveform Style Editor you can modify these characteristics. The Waveform

Style Editor settings are saved in the configuration file. You can also modify the waveform color by modifying the resource variables. See Chapter 16, VirSim Setup for more information about modifying resources.

Table A-2 VHDL Defaults

Signal Value	Encoding Type	Default Color	Waveform Display	Resource Variable
STD_Logic	"U’	Gray	Filled box	dsiColor03
	’X’	Red	Filled box	dsiColor23
	’0’	Blue	Line at bottom	dsiColor13
	’1’	Green	Line at top	dsiColor18
	’Z’	Gold	Line at midpoint	dsiColor06
	’W’	Red	Filled box	dsiColor03
	’L’	Blue	Line at bottom	dsiColor13
	’H’	Green	Line at top	dsiColor18
	’-’	Brown	Filled box	dsiColor24
Bit	’0’	Blue	Line at bottom	dsiColor13
	’1’	Green	Line at top	dsiColor18
BOOLEAN	FALSE	Blue	Line at bottom	dsiColor13
	TRUE	Green	Line at top	dsiColor18
Other scalar types	Any vale	Blue‘	Open box with value	dsiColor13
All vector types	Any value	Aqua	Open box with value	dsiColor13

Waveform Style: EPIC Defaults

The following tables describe rules for waveform display of EPIC values in the Waveform Window. The value strings also apply to the Register Window. The Waveform Style Editor identifies signals by the Signal Value listed in the table.

Table A-3 EPIC Scalar Signal Values

Scalar Value	Signal Type	Signal Value	Waveform Display
0	Logic zero (strong drive)	Strong ZERO	Blue line at bottom of waveform
1	Logic one (strong drive)	Strong ONE	Green line at top of waveform
U	Undefined (strong drive)	Strong UNDEF	Solid red box
L	Logic zero (high impedance)	Hi-Z ZERO	Gold line at bottom of waveform
H	Logic zero (high impedance)	Hi-Z ONE	Gold line at top of waveform
X	Logic zero (high impedance)	Hi-Z UNDEF	Solid gold box
N	Logic zero (driving bi-directional port)	Biput ZERO	Blue line at bottom of waveform
T	Logic zero (driving bi-directional port)	Biput ONE	Green line at top of waveform
Y	Logic zero (driving bi-directional port)	Biput UNDEF	Solid red box
?	Logic zero (no value set for signal)	UNDEF	Solid gray box

Table A-4 EPIC Vector Waveforms

Vector Value	Signal Type	Signal Value	Waveform Display
All unset	All ?	All UNSET	Solid gray box
All high impedance	All L,H, or X	All Hi-Z	Solid gray box
Some undefined	All U or Y	Some UNDEF	Solid red box
Some unset	At least one ? and any other value	Some UNSET	Gray open box with string
Some high impedance	‘No ?, at least one L, H, or X and any other value	Some Hi-Z	Gold open box with string
Some undefined	No ?, L, H, or X. At least one U or Y and any other value	Some UNDEF	Red open box with string
All 0’s and 1’s	0, 1, N, or T	All data	Aquamarine open box with string

Table A-5 EPIC Vector Value Strings

Vector Value	Signal Value	Waveform Display
Unset	All or some bits ?	?
All bits high impedance	All H, L, or X	Z
Some high impedance	No ?. At least one L, H, or X and any other value	U
Undefined	No ?. Some or all U	U
All 0’s and 1’a	All 0, 1, N, or T	0 through f

Index

Symbols

-I 2-4
-PP 2-4
-RPP 2-6
$vcdplusautoflushoff 17-6
$vcdplusautoflushon 17-6
$vcdplusdeltacycleoff 17-21
$vcdplusdeltacycleon 17-20
$vcdplusevent 17-23
$vcdplusflush 17-6
$vcdplusglitchoff 17-22
$vcdplusglitchon 17-22
$vcdplusoff 17-5
$vcdplustraceoff 17-19
$vcdplustraceon 17-19
+simargs 2-10, 2-15
+vcdfile 2-6
+vpdbufsize 17-25
+vpddrivers 17-26
+vpdfile 17-25
+vpdfilesize 17-25
+vpdignore 17-26
+vpdnocompress 17-27
+vpdnostrengths 17-27
+vpdports 17-27

A

Adding signals to groups 3-9
Align Command 7-18
assertion files 19-3
Assertions, OpenVera 19-1
Assignment Statements 6-6
audience i-xxiv
AutoGroups 4-13

B

benefits of OpenVera Assertions 19-2
binarySpacing 16-6
Breakpoint Group icon 8-20
Breakpoints
  Clearing 5-12
  Expression 8-11
  Line 8-14
  Setting 5-11
  Statement indicators 5-3
Building Buses 13-1
Bus Builder 13-1
Button file iwbuttons 8-16
  Selection substitution 8-19

C

C1 Cursor 4-5, 4-16
  Searching with expressions 15-11
C2 Cursor 4-16
Clear Breakpoints Command 5-12, 5-23
cnt.txp 19-3
Colormap 16-17
Configuration file
  Command line, loading from 2-3, 2-10
  Format of 2-29
  Incremental loading of 2-30
  Nesting configuration files 2-30
  Saving and loading 2-26
Context-Sensitive Menus, See CSM
CSM
  Format 7-14
  Label 7-13
  Logic Browser 6-23
  Module Instance 6-25
  Port Instance 6-26, 6-28
  Radix 7-12

D

decimalSpacing 16-6
Defining buttons 8-16
Delta Cycle Information
  Capturing delta cycle information 17-20
Design top 9-4
Designator 7-8
Dialogs
  Expressions Dialog 15-2
  Markers Dialog 14-3
  Radix Dialog 11-1
  Simulator Invocation Dialog 8-6
  Time Scale Dialog 10-1
  View Editor Dialog 7-4
Display Command 7-19
Display Unit, configure 10-2
Distribute Command 7-19
Drivers
  Signal drivers in Waveform Window 4-47

E

Edge-triggered expressions 15-13
Edit Parent Command 5-23
Encapsulated PostScript 4-42
EPIC
  Waveform display characteristics A-6
EVCD
  Converting to VCD+ 18-2
Event Origin
  Classifications 12-7
  Debugging 12-8
example
  temporal assertion file 19-3
Expressions 15-1
  Creating expressions 15-7
  Displaying expressions 15-9
  Edge-triggered searches 15-13
  Entering signal names 15-6
  Expressions Dialog 15-2
  Level sensitive searches 15-12
  Requirements 15-2
  Searching 15-11
  Trigger Types 15-10
  Updating expressions 15-8
Expressions Dialog 15-2

F

facilities, test 19-2
files, temporal assertion 19-3
flow 19-5

G

Go icon 8-20
Grouping signals 3-9
Groups 4-11– 4-13

H

help

Synopsys Technical Support Center i-xxvii
hexSpacing 16-6
Hierarchical Resolution 6-8
Hierarchy
navigation 3-6
Hierarchy Browser
Adding Signals to Groups 3-9
Definition 1-9
Menu Bar 3-13
History file
Closing from GUI 2-16
Displaying open 3-6
Multiple files, opening 2-23
Opening from GUI 2-16
Reopen after updating 3-6
Single file, open 2-22
Switching between open designs 3-6
HR 6-8

I

-I 2-4
Ignoring Calls and License Checking 17-26
information, other sources i-xxv
Instance Group Edit Dialog
Options 5-14
Instance Groups 5-13
Interactive Window 8-1
$timeformat 8-4
button file iwbuttons 8-16
Command prompt 8-2
Continuing simulation 8-20
Defining buttons 8-16
Definition 1-8
Displaying data 8-20
Edit commands 8-21
File commands 8-21
History pane 8-2
Invoking 8-6
Menu Bar commands 8-21
Setting breakpoints 8-9
Sim Commands 8-22
Simulation command line 8-8
Simulator controls 8-3
Simulator Invocation Dialog 8-6
Status 8-5
Tool Bar 8-20
User-defined buttons 8-3
Window areas 8-2
introducing OpenVera Assertions 19-2

K

Key Terms and Concepts 1-4

L

Level sensitive expressions 15-12
Link
Linking windows 1-21
Link icon 8-20
Loading configuration files 2-26
Logic Browser
Connection Dialog 6-14
Context-Sensitive Menus 6-23
Controlling simulation 8-15
Definition 1-14
Graphical objects 6-5
Point-and-click navigation 6-10
Previous View/Next View 6-21
Setting display text 6-19
Single net
Multiple net 6-30
Tool Bar 6-21
Verilog Source Compiler 6-4

M

manual
audience i-xxiv
related publications i-xxv
Mapping List Editor 11-5
Markers
Creating 14-3, 14-5
Editing 14-3, 14-6
Linking windows 1-21
Setting 14-6
Markers Dialog 14-3

Multiple nets in Logic Browser 6-30

N

NA 6-8
NA text 6-8
Navigating a design
  Hierarchy navigation 3-6
  Searching for signals 3-7
  Selecting Signal Range 1-18
Next View 6-21
NL 6-7
numericSeparator 16-6

O

octalSpacing 16-6
OpenVera Assertions
  benefits 19-2
  flow 19-5
  introduction 19-2
  overview 19-3
OpenVera Assertions, viewing 19-1
OVA, see OpenVera Assertions
-ova_debug option 19-5
-ova_debug_vpd option 19-5

P

-PP 2-4
Precision, configure 10-3
Previous View 6-21
Project Window 9-1
  Active library 9-5
  Adding files 9-33
  Adding libraries 9-5
  Analyze options 9-41
  Command line, executing commands 9-47
  Commands 9-48
  Constants, creating and modifying 9-36
  Default library 9-5
  Default settings, changing 9-39
  Defining projects and workspaces 9-1
  Definition 1-15
  Dependency folder view 9-14
  Design top folder view 9-11
  Directory, selecting 9-35
  Elaborate options 9-42
  Files and directories 9-54
  Library folder view 9-15
  Library view 9-18
  Makefiles 9-56
  Moving projects 9-55
  Opening files 9-31
  Output Pane 9-20
  Project folder view 9-10
  Simulate options 9-44
  Source file folder view 9-12
  Tearing off windows 9-20
  Verilog compile options 9-45
  Verilog libraries, creating 9-38
  Verilog run options 9-46
  View pane 9-9
  Workspace directory
    Project directory
      Project library 9-54
  Workspace folder view 9-9
  Workspace pane 9-5
  Workspace, creating 9-29

R

Radix
  editing radix 11-4
  editor
    user defined 11-3
  mapping 11-5
  selecting in Waveform window 4-5
  using a radix 11-1
Register Window 7-1
  Align Commands 7-18
  Aligning objects 7-7, 7-8
  Controlling simulation 8-15
  Designator 7-8
  Display Command 7-19
  Display menu 7-8
  Distribute Commands 7-19
  Distributing objects 7-8
  Edit commands 7-17

File Commands 7-16
Graphics Menu 7-18
Menu Bar commands 7-15
Signal properties CSM 7-9
Tool bar 7-9
Two-button mouse 7-1
View Editor Dialog 7-4
Views Command 7-17
Window areas 7-1
results 19-5
-RPP 2-6

S

Search pane 3-2
Searching
Events, searching for 4-44
Filters, using 3-7
Signal values, searching for in Wave Window 4-17
Signals, searching for 3-7
Using wildcard characters 3-7
sequence of events, describing 19-2
Setup
Common Settings 16-4
VirSim Platforms 16-2
Signal groups
Adding signals 3-9
Signal Name pane 4-2, 4-4
Signal Select pane 3-2
Signal Value pane 4-4
Signals
AutoGroups 4-13
Expanding or collapsing waveform vectors arrays, records, or busses 4-13
Loading view 6-7
Reordering signals for waveforms 4-10
Searching for 3-7
Types 4-5
Using groups 4-11
Verilog waveform characteristics A-2
Waveform color styles A-2
Simulation
Controlling simulation
From Logic Browser 8-15
From Register Window 8-15
From Waveform Window 8-12
Invoking Interactive Window 8-6
Simulation command line 8-8
Simulator controls
Scope control 8-4
Step controls 8-3
Time display 8-4
Simulator Invocation Dialog 8-6, 8-7
Single nets in Logic Browser 6-30
Source Window 5-1
Clear Breakpoints Command 5-23
Clearing Breakpoints 5-12
Control Panel 5-5
CSM commands 5-20
Defining markers 5-16
Definition 1-13
Display menu 5-24
Edit Menu 5-23
Edit Parent Command 5-23
Edit Source Command 5-23
Entering a scope 5-4
Execution pane 5-3
File Menu 5-22
Instance Groups 5-13
Menu Bar commands 5-22
Setting breakpoints 5-11
Show Execution Command 5-24
Source Text pane 5-4
Tool bar 5-18
Window areas 5-2
Sources
In Logic Browser 6-4
Loading 2-31
Use Sources 2-19
Starting VirSim 2-1
Configuration files, loading from GUI 2-26
History files, loading from GUI 2-16
Opening history files from GUI 2-16
Overview 2-7
Scirocco, starting VirSim from 2-7
Standalone Installations, starting Virsim from 2-11
VCS, starting VirSim from 2-2
Creating history files for 2-3

Interactive mode 2-2
Post-processing mode 2-5
VCD+ files 2-3
Stop icon 8-20
support
Synopsys Technical Support Center i-xxvii

T

temporal assertion files 19-3
test facilities 19-2
The 2-17
Time Scale Dialog 10-1
Configuring Precision 10-3
Time Units 10-1– 10-3
timeSpacing 16-6
Translating VCD and VCD+ 18-1
Trigger types 15-10

U

Undo Command 7-17
Unique Events
$vcdplusevent 17-23
$vcdplusglitchoff 17-22
$vcdplusglitchon 17-22
Update Icon 8-20
Use Sources 2-19
User-Defined Buttons, Register Window 8-3

V

VCD
Converting to VCD+ 18-2
VCD+ 17-2
Advantages 17-2
Capturing data 17-3
Closing from GUI 2-16
Command line arguments 2-9, 2-14
Command line options
Buffer size 17-25
Bypass data compression 17-27
Control maximum file size 17-25
Do not store strength information 17-27
Ignore $vcdplus calls in code 17-26
Set output file name 17-25
Store driver information 17-26
Store port information 17-26
Conversion from VCD 18-2
Managing simulation 17-27
Opening from GUI 2-16
System Tasks
$vcdplusautoflushoff 17-6
$vcdplusautoflushon 17-6
$vcdplusdeltacycleoff 17-21
$vcdplusdeltacycleon 17-20
$vcdplusevent 17-23
$vcdplusflush 17-6
$vcdplusglitchoff 17-22
$vcdplusglitchon 17-22
$vcdplusoff 17-5
$vcdpluson
vcd2vpd command 18-2
+vcdfile 2-6
vectorSeparator 16-6
verify 19-5
Verilog 2-13
System Tasks
$vcdplustraceoff 17-19
$vcdplustraceon 17-19
Waveform display characteristics A-2
View Editor Dialog 7-4
Views
Designing 7-1
Views Command 7-17
VirSim 19-5
VirSim platforms 16-2– 16-3
vpd2vcd command 18-5
VSC 6-4

W

warnIfConfigurationModified 16-7
Waveform pane 4-6

Waveform Window
Controlling simulation 8-12
Debugging Event Origin 12-8
Definition 1-11, 4-2
Drivers in 4-47
Edit Menu Commands 4-51
Expanding and collapsing vectors, arrays, records, or busses 4-13
Expressions, using 15-1
File Menu Commands 4-49, 4-50
Groups
4-11
Creating AutoGroups 4-13
Manually creating 4-12
Switching between multiple groups 4-13
Positioning Cursor C1 and C2 4-16
Reordering signals 4-9, 4-10
Searching for a signal value 4-17
Selecting signals 4-9
Signal groups 4-11
Signal Name pane 4-2, 4-4
Signal Value pane 4-2, 4-4
Status bar 4-8
Tool Bar 4-43
Load icon 4-43
Search icons 4-44
Zoom In 4-43
Zoom Percent icon 4-44
Waveform height 4-31
Waveform pane 4-2, 4-6
Zoom Commands 4-52
Zooming waveforms 4-14
Waveforms
EPIC display characteristics A-6
Grouping signals 4-11
Modifying height 4-31
Printing 4-38
Searching for events 4-44
Verilog display characteristics A-2
Windows NT™ 16-3

X

X Resources 16-3
Color model 16-16

Z

Z1 2-31
Zoom
Commands 4-52
Zoom Percent icon 4-44